Relational Databases

Lecture 1

Database Systems

- A database system consists of
  - Data
  - Software
  - Hardware
  - Users
- We focus mainly on the software

- Database systems allow users to
  - Store
  - Update
  - Retrieve
  - Organise
  - Protect

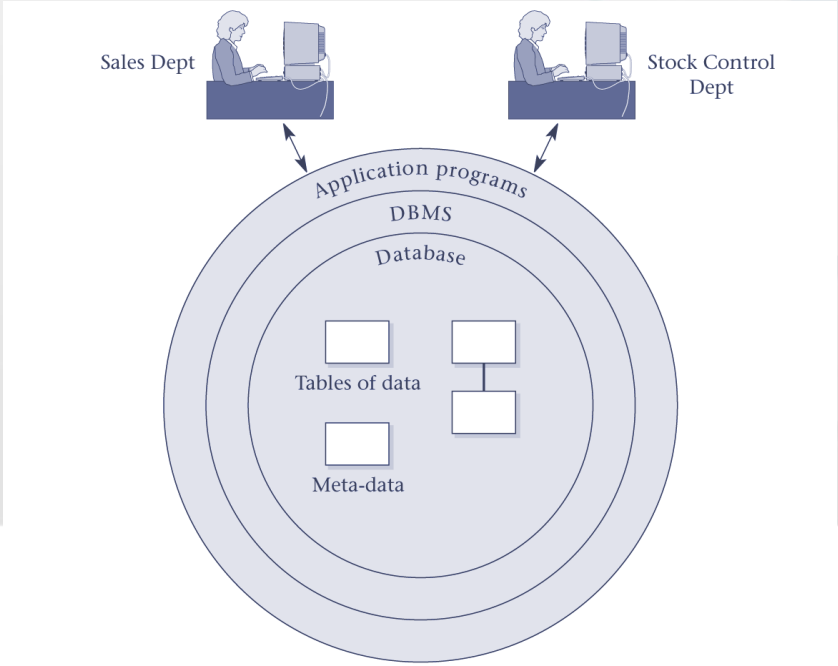
  their data.

Examples of Database Systems

- Purchases from the supermarket
- Purchases using your credit card
- Booking a holiday at the travel agents
- Using the local library
- Renting a video
- Using the Internet

Database Management Systems (DBMS)

- A database is a collection of information
- A database management system (DBMS) is the software than controls that information



- Examples:
  - Oracle
  - DB2 (IBM)
  - MS SQL Server
  - MS Access
  - FoxPro
  - PostgreSQL
  - MySQL

Database Management system

A general-purpose database management system (DBMS) is a software system designed to allow the definition, creation, querying, update, and administration of databases.

Components of DBMS Environment

Hardware

- Can range from a PC to a network of computers.

Software

- DBMS, operating system, network software (if necessary) and also the application programs.

Data

- Used by the organization and a description of this data called the schema.

Procedures

- Instructions and rules that should be applied to the design and use of the database and DBMS.

People

- Includes database designers, DBAs, application programmers, and end-users.

What the DBMS does

- Provides users with
  - Data definition language (DDL)
  - Data manipulation language (DML)
  - Data control language (DCL)
- Often these are all the same language

- DBMS provides
  - Persistence
  - Integrity
  - Security
  - Data independence
- Data Dictionary
  - Describes the database itself

File Based Systems

- File based systems
  - Data is stored in files
  - Each file has a specific format
  - Programs that use these files depend on knowledge about that format
- Problems:
  - No standards
  - Data duplication
  - Data dependence
  - No way to generate ad hoc queries
  - No provision for security, recovery, concurrency, etc.

Relational Systems

- Problems with early databases
  - Navigating the records requires complex programs
  - There is minimal data independence
  - No theoretical foundations
- Then, in 1970, E. F. Codd wrote “A Relational Model of Data for Large Shared Databanks” and introduced the **relational model**

Relational Systems

- Information is stored as *tuples* or *records* in *relations* or *tables*
- There is a sound mathematical theory of relations
- Most modern DBMS are based on the relational model

- The relational model covers 3 areas:
  - Data structure
  - Data integrity
  - Data manipulation
- More details in the next lectures…

Classical relation database follow the ACID Rules

Atomic : A transaction is a logical unit of work which must be either completed with all of its data modifications, or none of them is performed.

Consistent : At the end of the transaction, all data must be left in a consistent state.

Isolated : Modifications of data performed by a transaction must be independent of another transaction. Unless this happens, the outcome of a transaction may be erroneous.

Durable : When the transaction is completed, effects of the modifications performed by the transaction must be permanent in the system.

Overview

A Real World Model

- **The Entity Relationship (ER) Model:**
  - An entity relationship model, also called an entity-relationship (ER) diagram, is a graphical representation of entities (which will become your tables) and their relationships to each other.

A Relational Model of Databases

ELEMENTS OF RELATIONAL MODEL OF DATA:

- **Relation**: Table with columns and rows
- **Attribute**: Named column of a relation
- **Domain**: Set of allowable values for one or more attributes
- **Tuple**: Row of a relation
- **Degree**: Number of attributes in a relation
- **Cardinality**: Number of tuples in a relation
- **Relational Database**: Collection of normalized relations with distinct relation names

Relational Algebra and Relational Calculus

- **Relational algebra** and **relational calculus** are formal languages associated with the relational model.
- Five basic operations in relational algebra: **Selection**, **Projection**, **Cartesian product**, **Union**, and **Set Difference**.
- Also have **Join**, **Intersection**, and **Division** operations, which can be expressed in terms of 5 basic operations.

SQL (Structured Query Language)

- SQL is an example of a **transform-oriented language**, or a language designed to use relations to transform inputs into required outputs. As a language, SQL has two major components:
  - **SQL DDL (Data Definition Language)** for defining the database structure and controlling access to the data;
  - **SQL DML (Data Manipulation Language)** for retrieving and updating data

Normalization

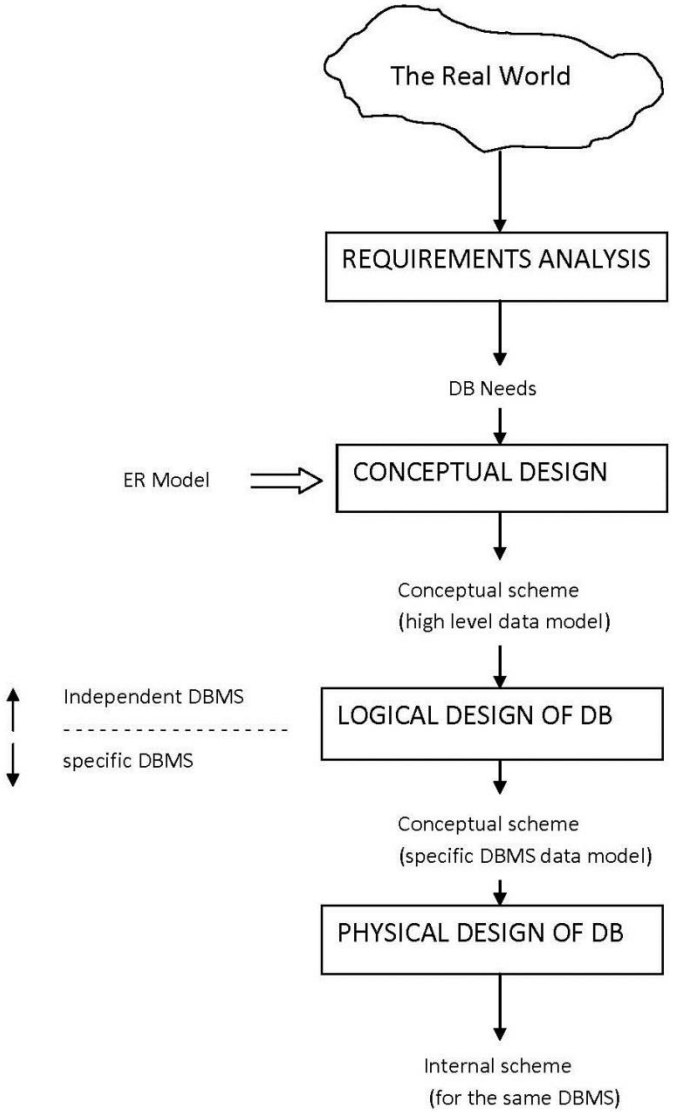
- **Normalization** is a technique for producing a set of relations with desirable properties, given the data requirements of an enterprise.
- The benefits of using a database that has a suitable set of relations is that the database will be easier for the user to access and maintain the data, and take up minimal storage space on the computer.
- **I, II, III, and IV NORMAL FORMS**!

Integrity

- **Integrity rules** are constraints or restrictions that apply to all instances of the databases (for example: integrity rules for the primary and foreign keys)

- Entity integrity
- Referential integrity
- General constraints

A Real World Model


The Real World
REQUIREMENTS ANALYSIS
DB Needs
ER Model
CONCEPTUAL DESIGN
Conceptual scheme
(high level data model)
Independent DBMS
specific DBMS
LOGICAL DESIGN OF DB
Conceptual scheme
(specific DBMS data model)
PHYSICAL DESIGN OF DB
Internal scheme
(for the same DBMS)

Database Design

- Before we look at how to create and use a database we'll look at how to design one
- Need to consider
  - What tables, keys, and constraints are needed?
  - What is the database going to be used for?

- Conceptual design
  - Build a model independent of the choice of DBMS
- Logical design
  - Create the database in a given DBMS
- Physical design
  - How the database is stored in hardware

Entity/Relationship Modelling

- E/R Modelling is used for conceptual design
  - **Entities** - objects or items of interest
  - **Attributes** - facts about, or properties of, an entity
  - **Relationships** - links between entities

- Example
  - In a University database we might have entities for Students, Modules and Lecturers. Students might have attributes such as their ID, Name, and Course, and could have relationships with Modules (enrolment) and Lecturers (tutor/tutee)

Entities

- Entities represent objects or things of interest
  - Physical things like students, lecturers, employees, products
  - More abstract things like modules, orders, courses, projects
- Entities have
  - A general type or class, such as Lecturer or Module
  - Instances of that particular type, such as Steve Mills, John Smith are instances of Lecturer
  - Attributes (such as name, email address)

Attributes

- Attributes are facts, aspects, properties, or details about an entity
  - Students have IDs, names, courses, addresses, …
  - Modules have codes, titles, credit weights, levels, …
- Attributes have
  - A name
  - An associated entity
  - Domains of possible values

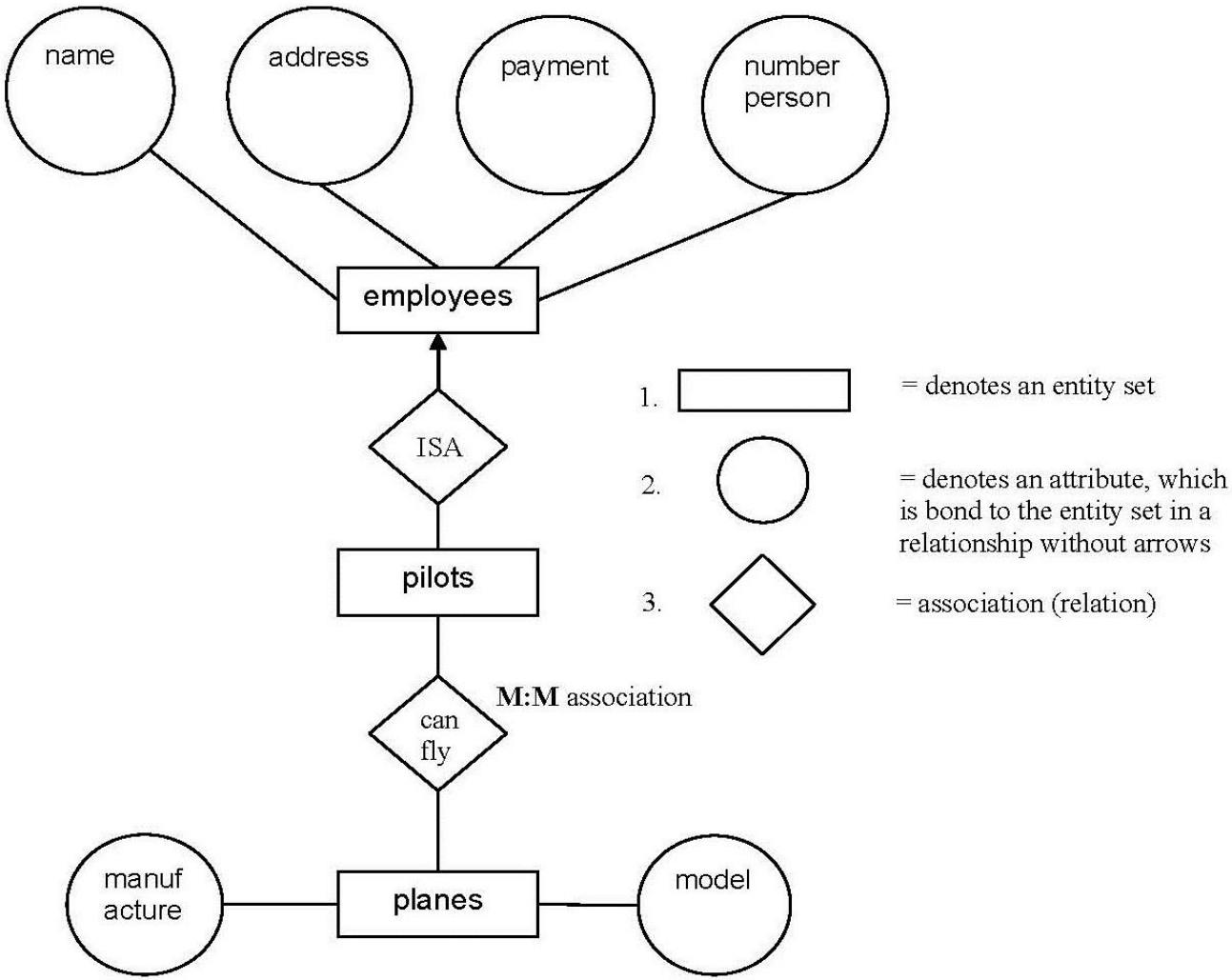
Relationships

- Relationships are an association between two or more entities
  - Each Student takes several Modules
  - Each Module is taught by a Lecturer
  - Each Employee works for a single Department
- Relationships have
  - A name
  - A set of entities that participate in them
  - A degree - the number of entities that participate (most have a degree 2)
  - A cardinality ratio

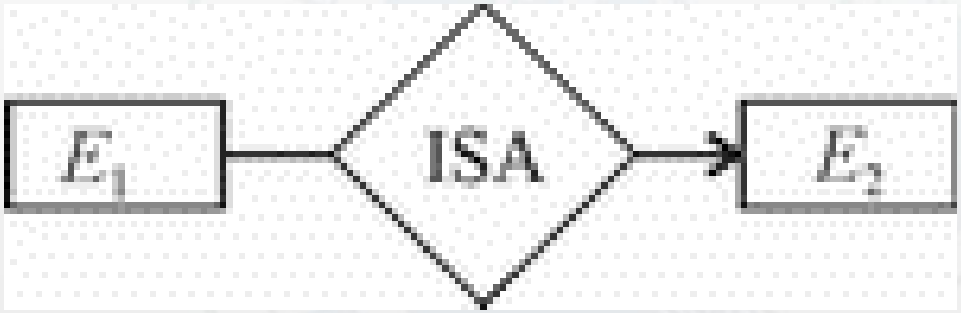
Cardinality Ratios

- Each entity in a relationship can participate in zero, one, or more than one instances of that relationship
- This leads to 3 types of relationship…

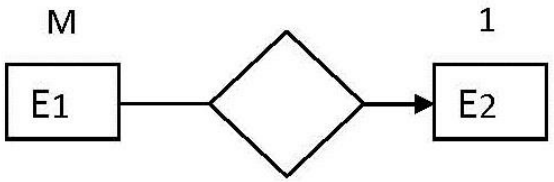
- One to one (1:1)
  - Each lecturer has a unique office
- One to many (1:M)
  - A lecturer may tutor many students, but each student has just one tutor
- Many to many (M:M)
  - Each student takes several modules, and each module is taken by several students

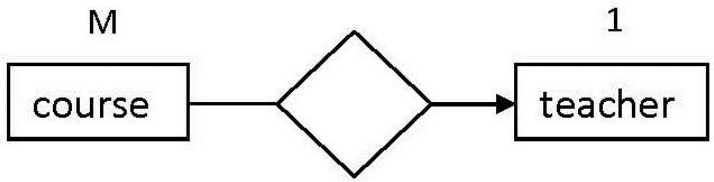

name
address
payment
number
person
employees
ISA
pilots
can
fly
M:M association
manuf
acture
planes
model
1.
= denotes an entity set
2.
= denotes an attribute, which
is bond to the entity set in a
relationship without arrows
3.
= association (relation)

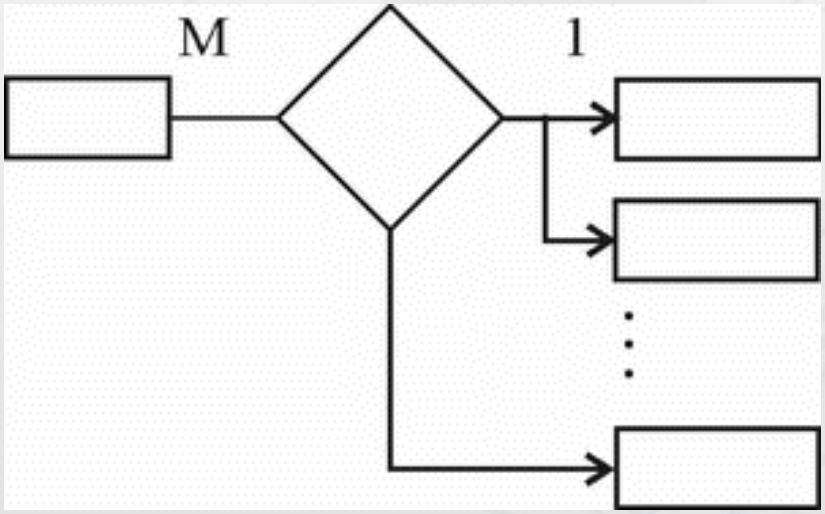
ISA association



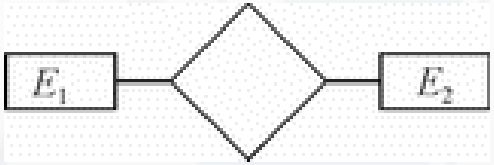
m:1 association

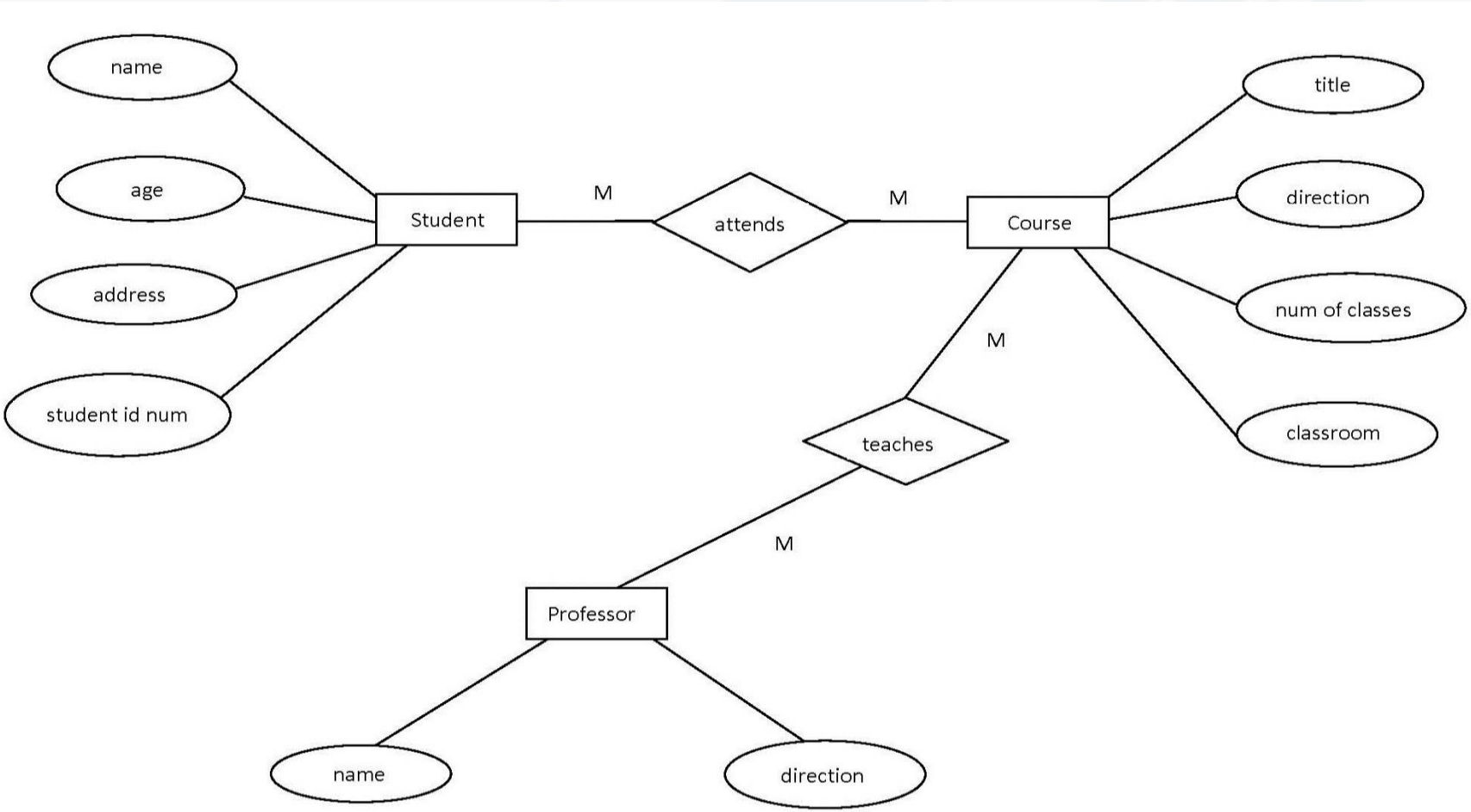




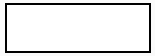
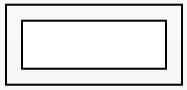
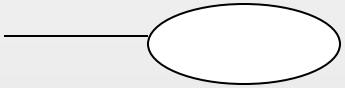
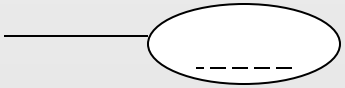
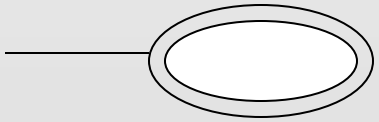


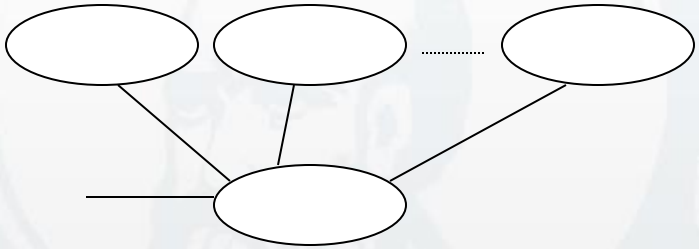
m:m association





Summary of ER diagram notation

SYMBOL	MEANING
	ENTYTY
	WEAK ENTITY
	RELATIONSHIP
	ATTRIBUTE
	KEY ATTRIBUTE
	MULTIVALUED

MEANING	SYMBOL
COMPOSITE ATTRIBUTE	
DERIVED ATTRIBUTE	

ER diagram for the COMPANY schema. With all role names included and with structural constraints on relationship specified using the alternate notation (min, max)