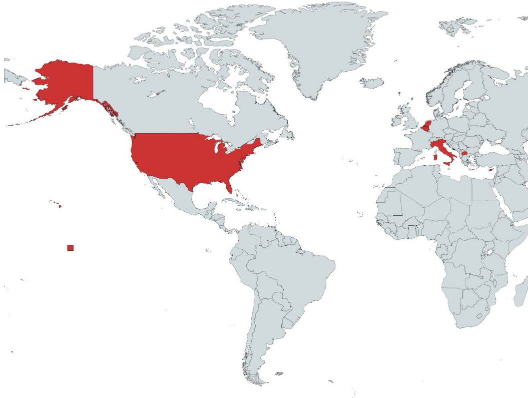
Databases 2022

Darko Bozhinoski,
Ph.D. in Computer Science

Agenda

- About Me
- General information about the course
- Data and Database Management Systems (DBMS)
- Database Design
- The Entity-Relationship Model

About Me



Lived and worked in
6 countries

Started Ph.D.

Ph.D. in Computer Science

Gran Sasso Science Institute, L'Aquila, Italy

Postdoc

IRIDIA, Artificial Intelligence, ULB, Brussels, Belgium

Researcher

Cognitive Robotics, TU Delft, Delft, Netherlands

Senior Researcher

FNRS, Belgium

Visiting Professor

InnoPolis, Russia

Nov 2013

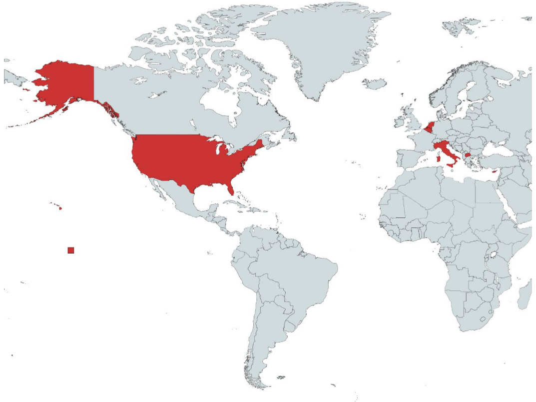
Dec 2017

Jan 2018

Jan 2019

Oct 2021

March 2022



Lived and worked in
6 countries

Research Focus

Autonomous systems

Model-driven engineering

Self-adaptive systems

Formal verification

Single and Multi-robot systems

Robot Swarms

Academic Partners

UNIVERSITÉ LIBRE DE BRUXELLES
IRIDIA
Institut de Recherches Interdisciplinaires et de Développements en Intelligence Artificielle

GSSI
GRAN SASSO SCIENCE INSTITUTE
SCHOOL OF ADVANCED STUDIES
Scuola Universitaria Superiore

TU Delft

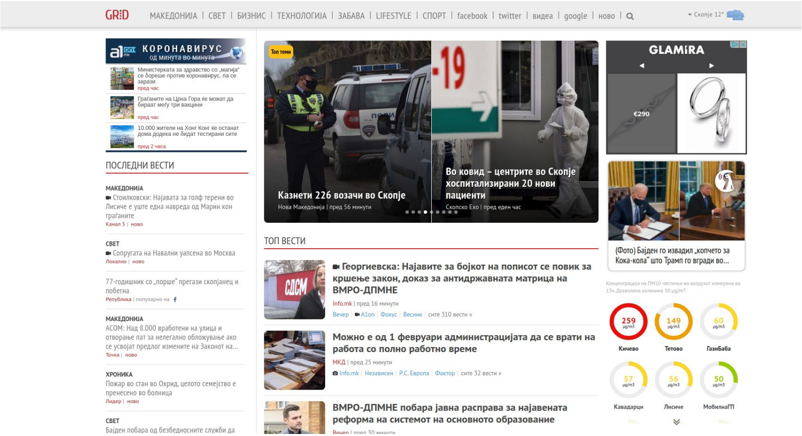
Carnegie Mellon University

UNIVERSIDAD POLITÉCNICA MADRID
POLITÉCNICA

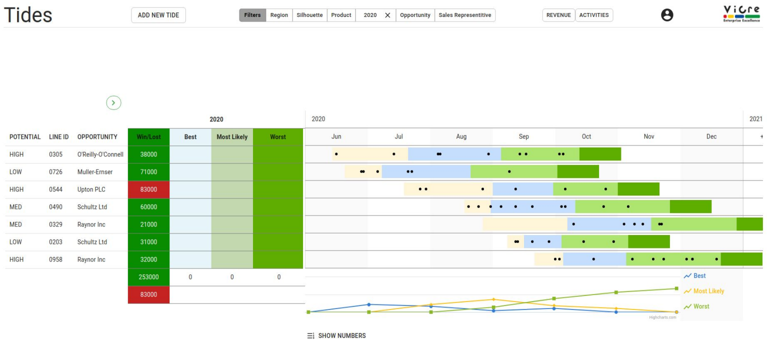
Universidad Rey Juan Carlos

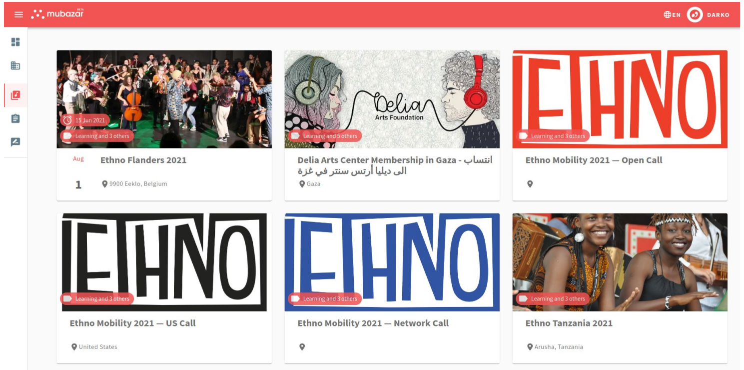
BOSCH
Invented for life

Industrial Projects



Collaboration with startups;





Information about the course

Team

Lectures:

Darko Bozhinoski (d.bozhinoski@innopolis.ru)

Tutorials:

Hamza Salem (h.salem@innopolis.ru)

Labs:

Darko Bozhinoski (d.bozhinoski@innopolis.ru)

Hamza Salem (h.salem@innopolis.ru)

Xavier Vasquez (x.vasquez@innopolis.university)

Materials and Communication

Moodle: [F2022 Databases 2022]

- Information about the course
  - Lecture Material
  - Tutorials and lab sessions with exercise
  - Information about assignments, labs
  - Information about your grades

- For urgent communication, we will use a Telegram group.
  Join the Telegram group for this course: https://t.me/+A4esblFdpsphMGQx

Required background knowledge

Previous attendance of Introduction to Software Engineering is considered necessary.

It is also recommended to pass Discrete Math/logic and in advance to be familiar with basic algebra and data structures.

Previous attendance of Introduction to Programming I is necessary to manage the fundamentals of programming.

Course Delivery

- Four weekly academic hours of lectures (two on Monday, two on Friday)
- Four weekly academic hours of tutorials (two on Monday, two on Friday)
- Four weekly academic hours of labs (two on Monday, two on Friday)
- For 8 weeks starting from March 14th

Self-study

Overall the course should take on average 15-18 hours per week of your life

- 9 hours are of classes, tutorials and labs
- The amount of study for the course is estimated in about 6 - 9 hours/week

Course Objectives

- Design, develop and implement a mid-scale relational database for an application domain using a relational DBMS
- Understand physical database design, implementation, and optimization issues
- Devise appropriate ways to store and index data, concurrent and safe access in data-intensive applications
- Design and maintain large distributed systems
- Use persistency tools in the context of modern software architectures

Assessment

- Mid-term Exam (20%);
- Final Exam (40%);
- Assignment (20%);
- Labs (20%);

Grading Model

- A -> [90, 100]
- B -> [75, 90)
- C -> [60, 75)
- D -> [0, 60)

In each lab, a student can get some scores*:

- 2 if implemented the correct solution of the task of the lab + attended
- 1 if attended but didn't achieve the right solution for the task
- 0 no attendance

* Students who got total score for all the labs will receive 20% in the total grade. All other values below this will be calculated accordingly.

Textbooks

- **Fundamentals of Database Systems.** Ramez Elmasri and Shamkant B. Navathe. Pearson.
- Raghu Ramakrishnan and Johannes Gehrke. **Database Management Systems (3rd Edition)**. McGraw-Hill (2003).
- Sam Lightstone, Toby Teorey and Tom Nadeau. **Physical Database Design - The Database Professional's Guide to Exploiting Indexes, Views, Storage, and More (1st Edition)**. Elsevier (2007).
- Abraham Silberschatz, Henry F. Korth, S. Sudarshan. **Database System Concepts (6th Edition)** McGraw-Hill (2010)

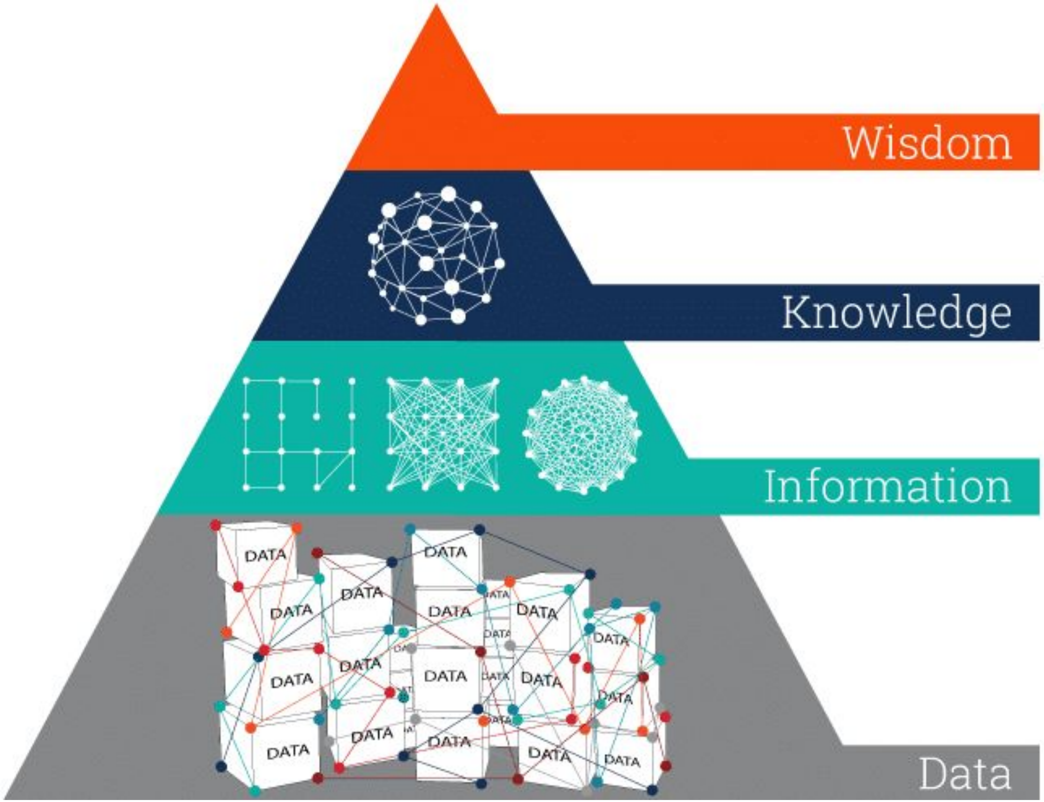
Introduction

Data, Database and Database Management Systems

- ❖ Data is the new currency.
- ❖ Data plays important role in the operational and organization level of big companies.

What is data?

- ❖ Data is collection of facts in a raw or unorganized form (e.g.,numbers or characters).



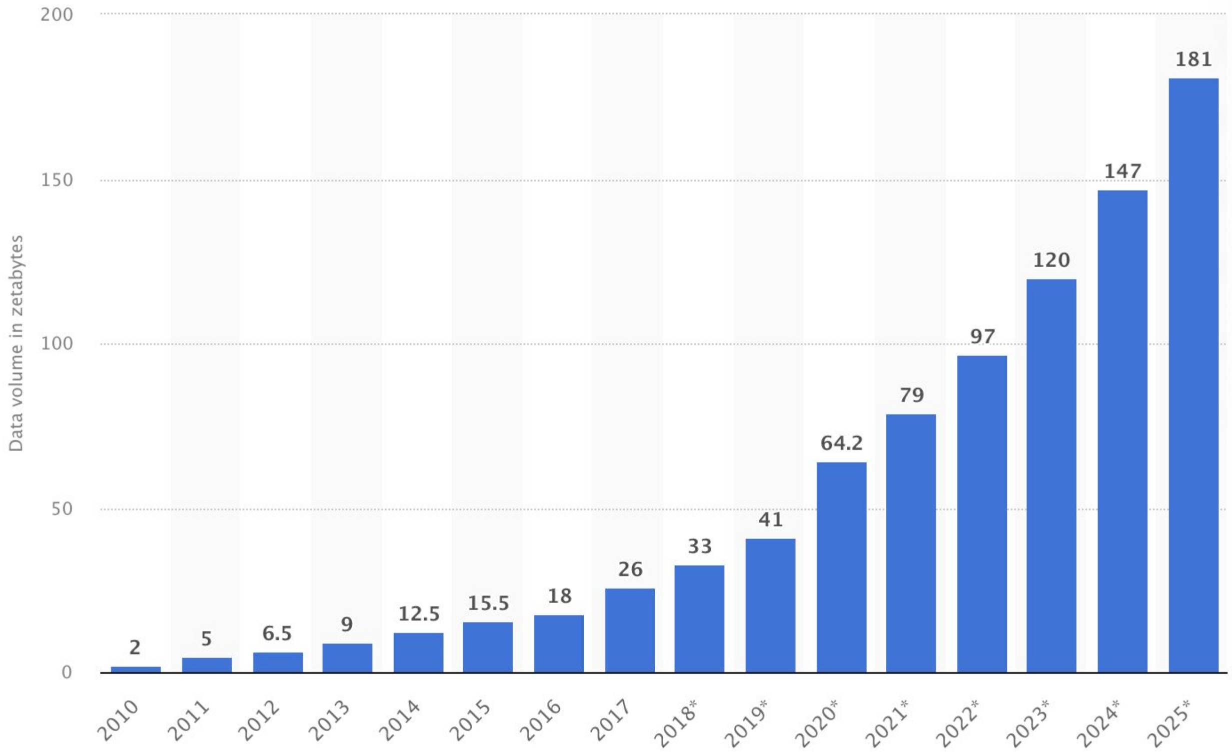
WHY? WHAT IS BEST?

HOW?

Each step up the pyramid answers questions about and adds value to the initial data.

Data, Information, Knowledge, Wisdom (DIKW) Pyramid

The amount of digital data in the world exceeded a zettabyte (10^21 bytes)



* Source Statista

Volume of data/information created, captured, copied, and consumed worldwide from 2010 to 2025

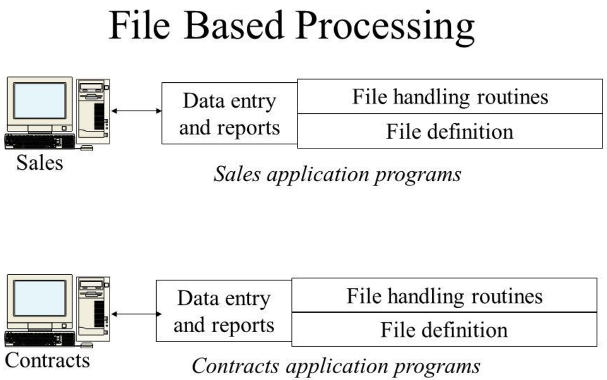
https://www.statista.com/statistics/871513/worldwide-data-created/

History of Databases

File-Based Systems

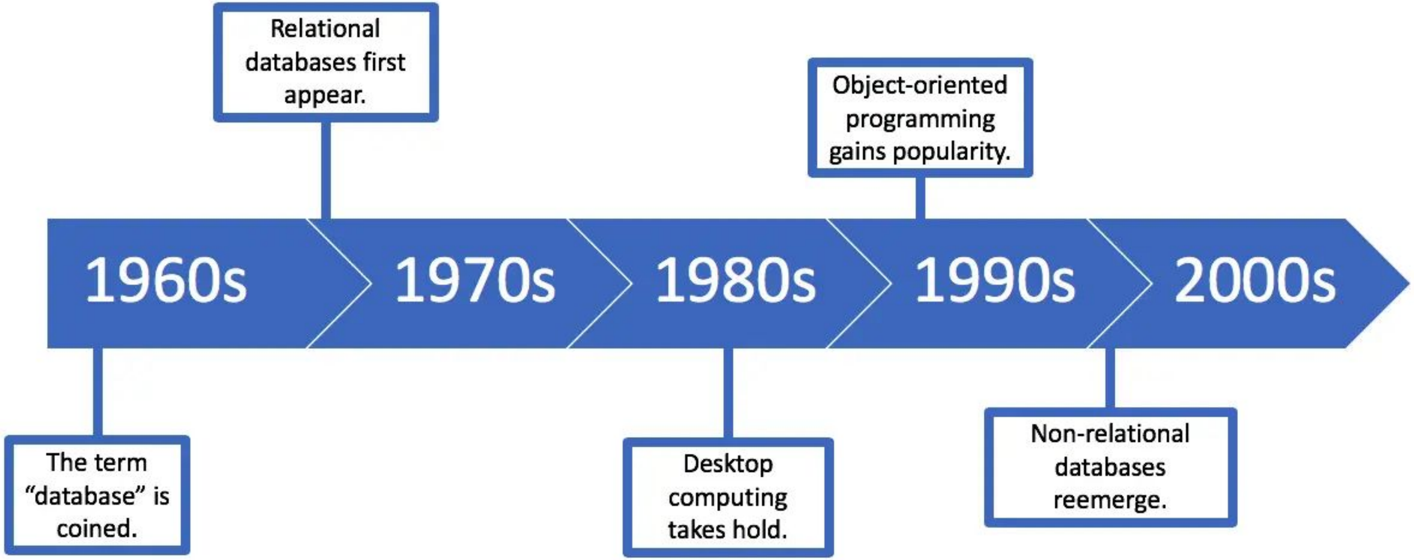
<u>Manual file systems:</u>

- Inadequate for processing of information in files;
- Inadequate for cross-reference;
- Lack of security;
- Prone to be damaged and misplaced
- Hard to make changes;
- Takes up a lot of physical space;



<u>File-Based Systems:</u>

- Predecessor to the DBMS
- A collection of application programs that perform services for end-users (e.g. production of reports)
- Each program defines and manages its own data


Relational databases first appear.
Object-oriented programming gains popularity.
1960s
1970s
1980s
1990s
2000s
The term “database” is coined.
Desktop computing takes hold.
Non-relational databases reemerge.

Database

- **Database** is a collection of related data.

- It describes the activities of one or more related organizations.

- E.g.A university database might contain information about the following:
  - Entities such as students, faculty, courses, and classrooms.
  - Relationships between entities, such as students' enrolment in courses, faculty teaching courses, and the use of rooms for courses

Database Management system (DBMS)

- A **database management system (DBMS)** is a computerized system that enables users to create and maintain a database.
  - designed to assist in maintaining and utilizing large collections of data
- DBMS is a *general-purpose software system* that facilitates the processes of *defining, constructing, manipulating,* and *sharing* databases among various users and applications.
- A **database-management system (DBMS)** is a collection of interrelated data and a set of programs to access those data.
- Primary goal of DBMS is to provide a way to store and retrieve database information that is both *convenient* and *efficient*.

DBMS Advantages over file-processing system

DBMS Advantages over file-processing system

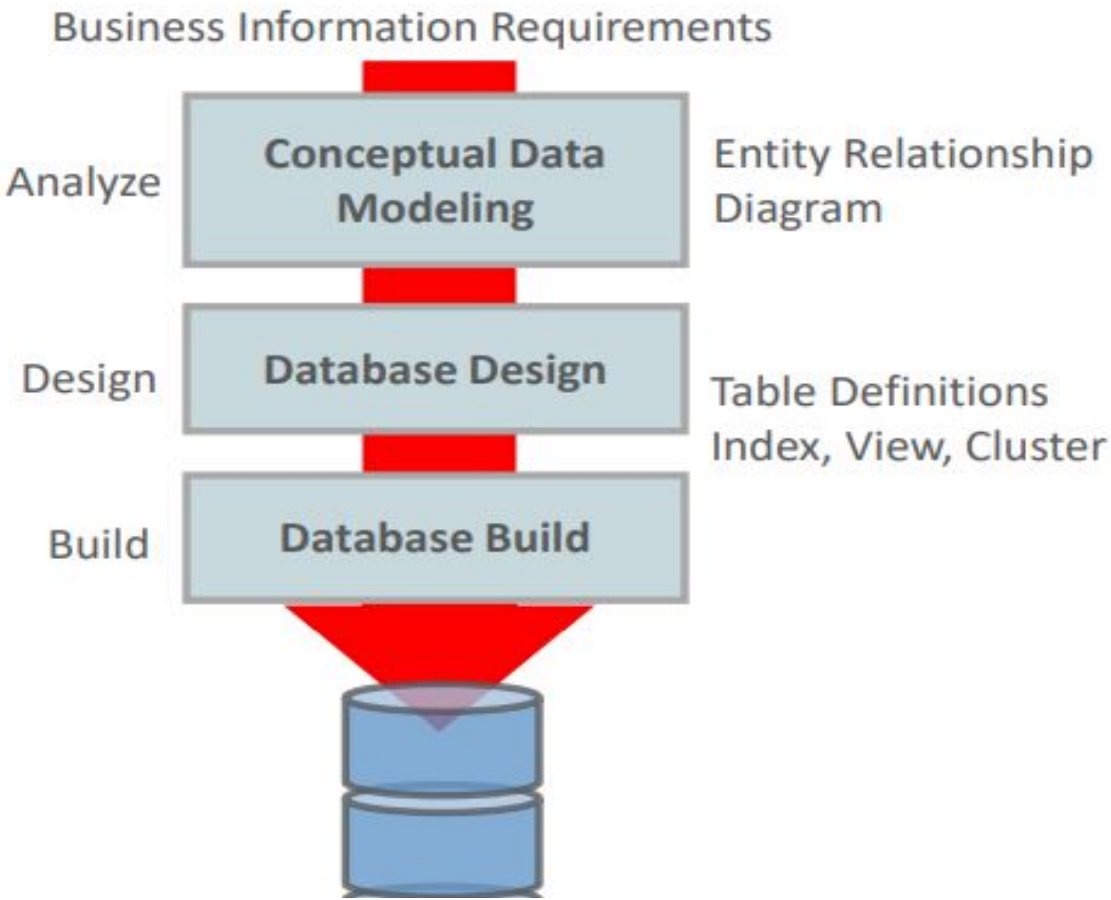
- **Data redundancy and inconsistency:** various files are likely to have different structures and the programs may be written in several programming languages. Moreover, the same information may be duplicated in several places (files).
  Difficulty in accessing data: conventional file-processing environments do not allow needed data to be retrieved in a convenient and efficient manner
- **Data isolation:** data are scattered in various files, and files may be in different formats, writing new application programs to retrieve the appropriate data is difficult.
- **Integrity problems:** consistency constraints.
- **Atomicity problem:** difficult to ensure atomicity in a conventional file-processing system.
- **Concurrent-access:** multiple users to access and update data simultaneously.
- **Security issues:** enforcing security constraints in file-processing system is difficult.

Database Design

Data Models

- ❖ A data model is an abstract model that organizes elements of data and standardizes how they relate to one another and to properties of the real-world entities
- ❖ A database model is a type of data model that determines the logical structure of a database and determines in which manner data can be stored, organized and manipulated
- ❖ Types of data models: Relational model; ER model, object-based data model, semistructured data model, etc.

Database Development Process



Design process

Requirements Engineering:

- Understanding what data has to be stored in the database
- What applications must be built on top of it
- What operations are most frequent and subject to performance requirements
- What the users want from the database

Data abstraction

Data abstraction refers to the suppression of details of data organization and storage, and the highlighting of the essential features for an improved understanding of data.

Conceptual level: a high-level description of the data to be stored and related constraints

Logical level: describes what data and relationships between data are stored.

Physical level: describes how the data are stored.(low-level data structures)

Conceptual Database Design

- Requirements analysis leads to a high-level description of the data

  Produced artifact: ER model

- The ER model is one of several data models used in database design

- **The goal is to create a simple description of the data that closely matches how users and developers think of the data**

Logical Database Design

Choosing a DBMS:
- a specific type with its own model;

Converting the conceptual database design into a database schema in the data model of the chosen DBMS

Output: Database schema;

Physical Database Design

❖ Consider the expected workloads that the database has to cope with

❖ Refining the database design to ensure that it **meets desired performance criteria** through different strategies**:**

Physical Database Design

- ❖ Consider the expected workloads that the database has to cope with
- ❖ Refining the database design to ensure that it **meets desired performance criteria** through different strategies**:**
  - ➢ Building indexes on some tables
  - ➢ Clustering some others
  - ➢ Redesign of parts of the database schema

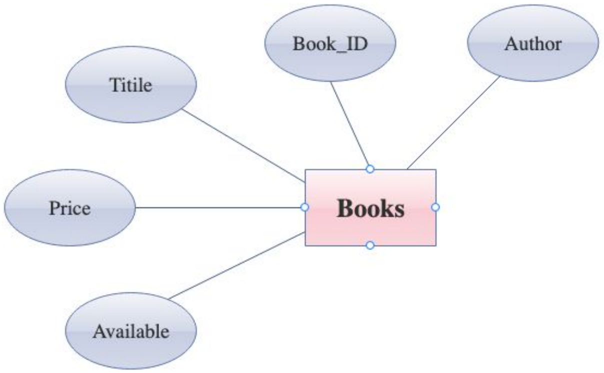
Entity-Relationship Model

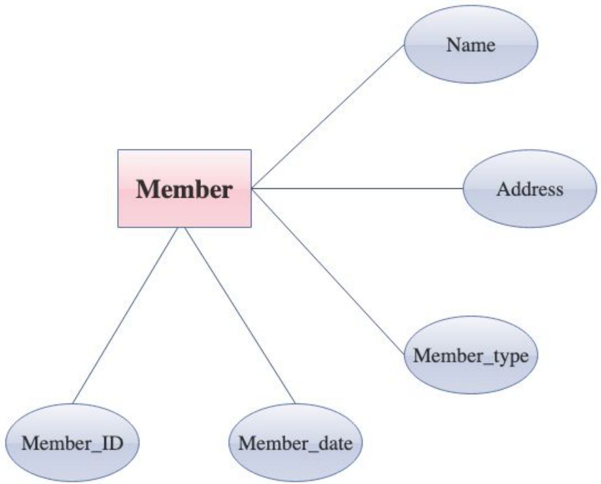
Basic concepts

- **Entity:** corresponds to a thing or object in the real world
- **Relationship:** corresponds to a link between entities
- **Attribute:** a property that describes an entity

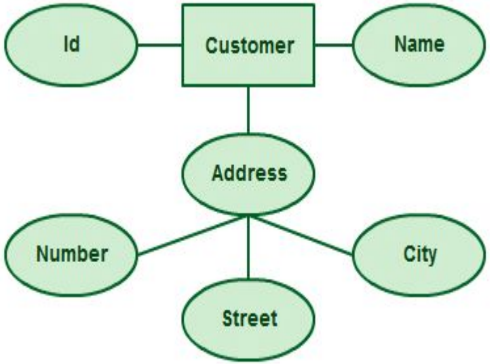
ER diagrams

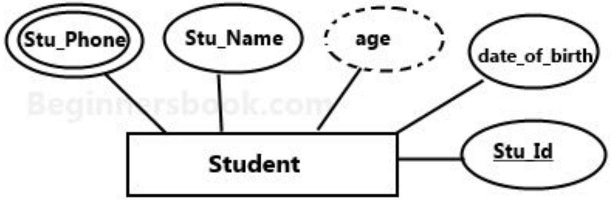
Entities have set a set of attributes.

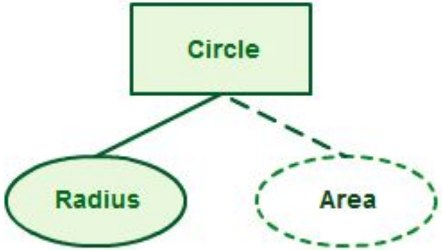




ER diagrams: Attribute Types







Multivalued attributes

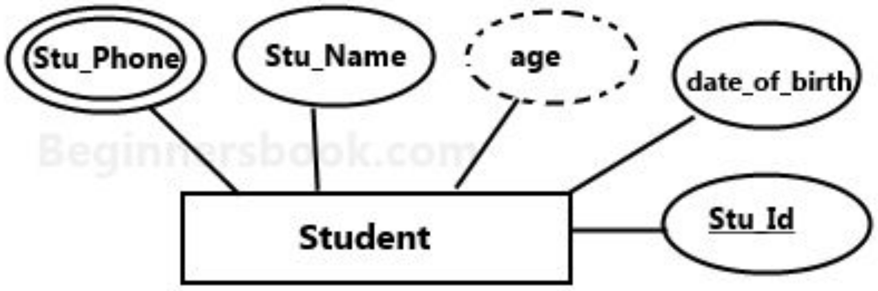
Composite vs. atomic attributes

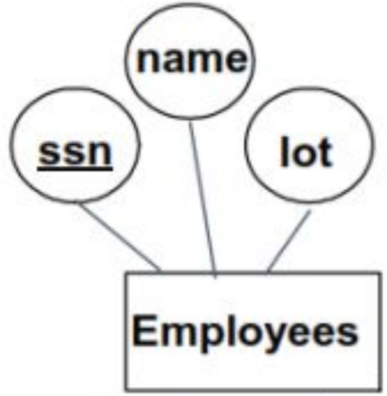
Stored vs. Derived attributes

ER diagrams: Key

Each entity set has a key:

<u>a set of attributes uniquely identifying an entity</u>





Entity Types and Entity Sets

Entity Types

- Describes a groups of entities that have similar characteristics/properties: For example, a university has thousands of students that share the same attributes, but each entity has its own value(s) for each attribute.
- An entity type defines a collection (or set) of entities that have the same attributes.
- Each entity type in the database is described by its name and attributes.

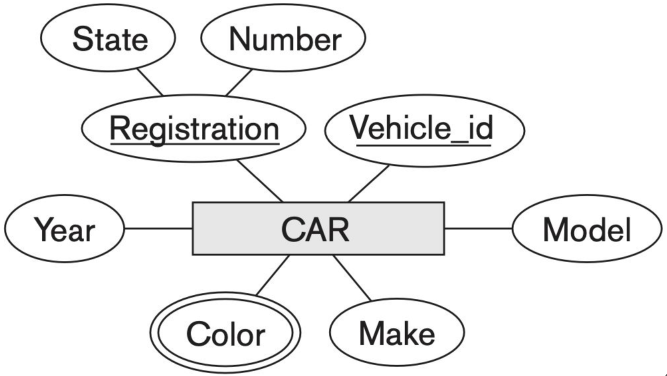
Entity Sets

- The collection of all entities of a particular entity type;

Entity keys

An attribute that uniquely defines an entity in an entity set:

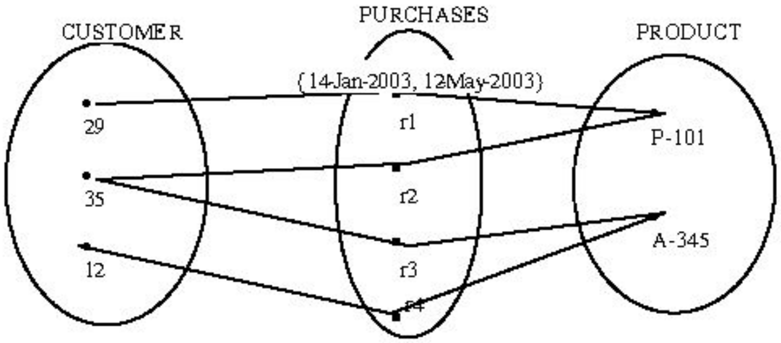
- **Super key:** A set of attributes (one or more) that together define an entity in an entity set
- **Candidate key:** A minimal super key
  - An entity set may have more than one candidate key
- **Primary key:** A candidate key chosen by the database designer to uniquely identify the entity set



ER diagrams: Relationships

- Relationship is specified whenever an attribute of one entity type refers to another entity type.

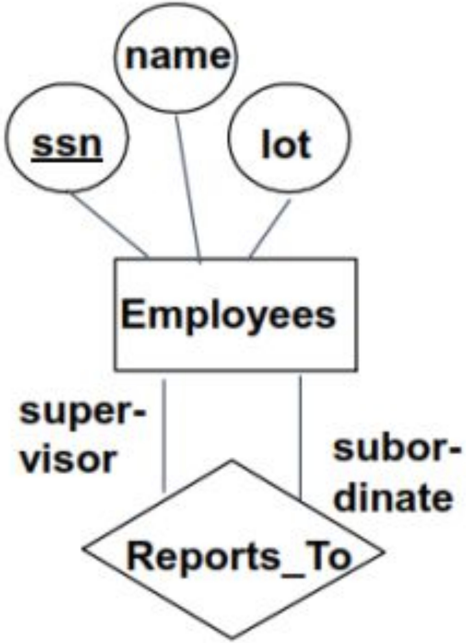
- Relationship Types/Sets/Instances
  - A **relationship type R** among n entity types $E_1, E_2, \ldots, E_n$ defines a set of associations - or a **relationship set** - among entities from these entity types. Both are referred with R.
  - Mathematically, the relationship set R is a set of relationship instances r_i, where each r_i associates n individual entities $(e_1, e_2, \ldots, e_n)$, and each entity e_j in r_i is a member of entity set E_j, $1 \le j \le n$.
  - Each entity type $E_1, E_2, \ldots, E_n$ <u>participates in the relationship type R</u>
  - Each individual entity $e_1, e_2, \ldots, e_n$ <u>participates in the relationship instance r_i</u> $= (e_1, e_2, \ldots, e_n)$.



ER diagrams: Relationships

- **Degree of a Relationship Type.** The degree of a relationship type is the number of participating entity types.

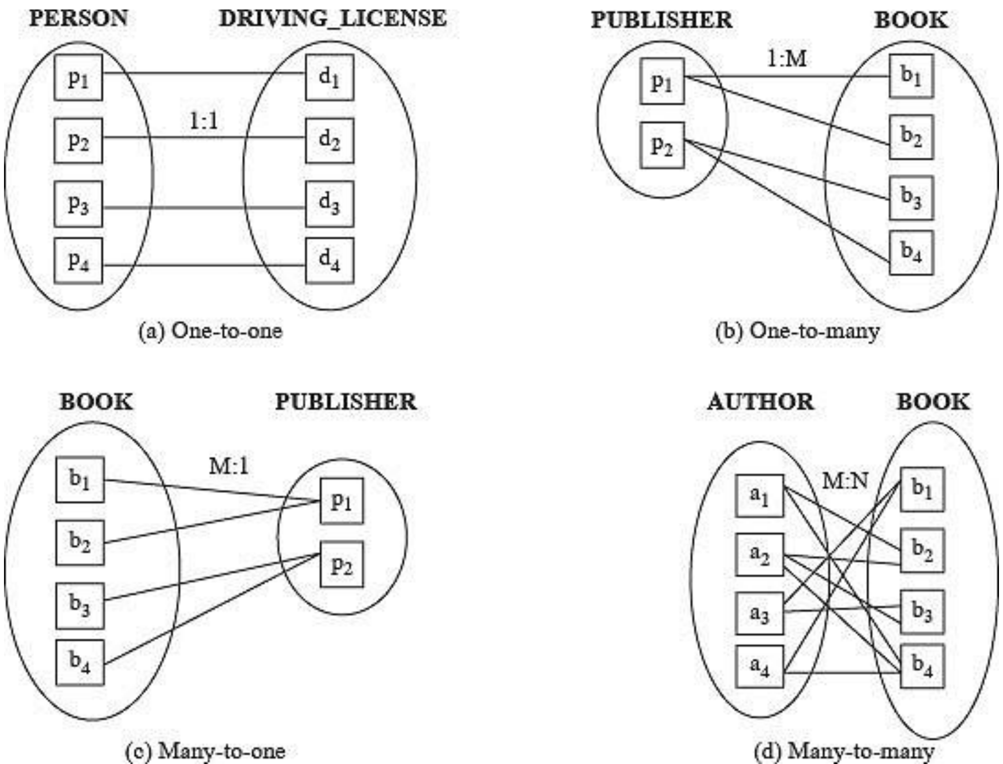
- The function that an entity plays in a relationship is called its role.
  - Roles are normally explicit and not specified. They are useful when a clarification is needed.
  - An entity could participate in different relationships, or in different “roles” in same relationship
  - **Recursive relationships:** If a same entity participates more than once



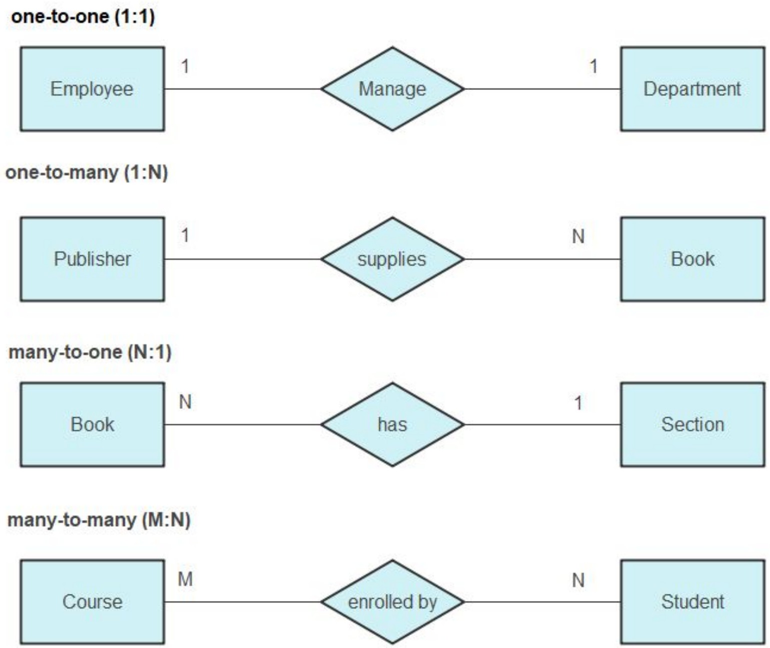
- Relationships can also have attributes;

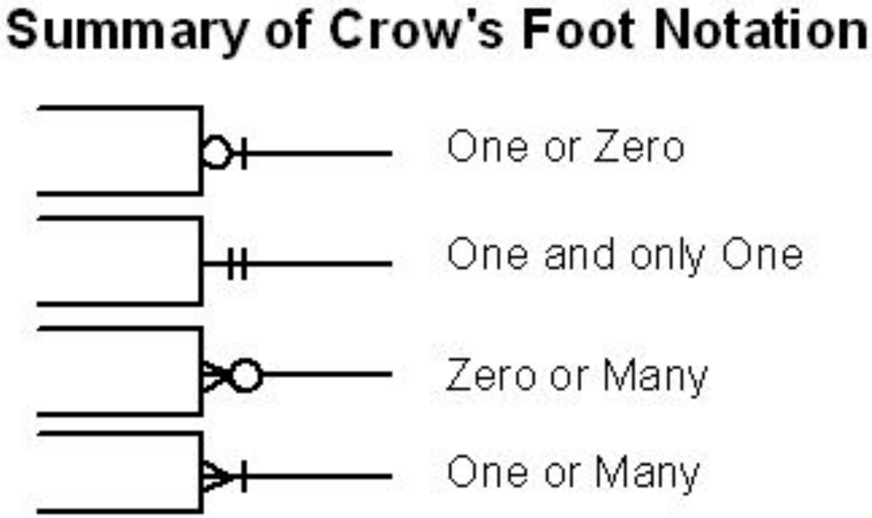
Cardinality ratios

Express the number of entities to which another entity can be associated via a relationship

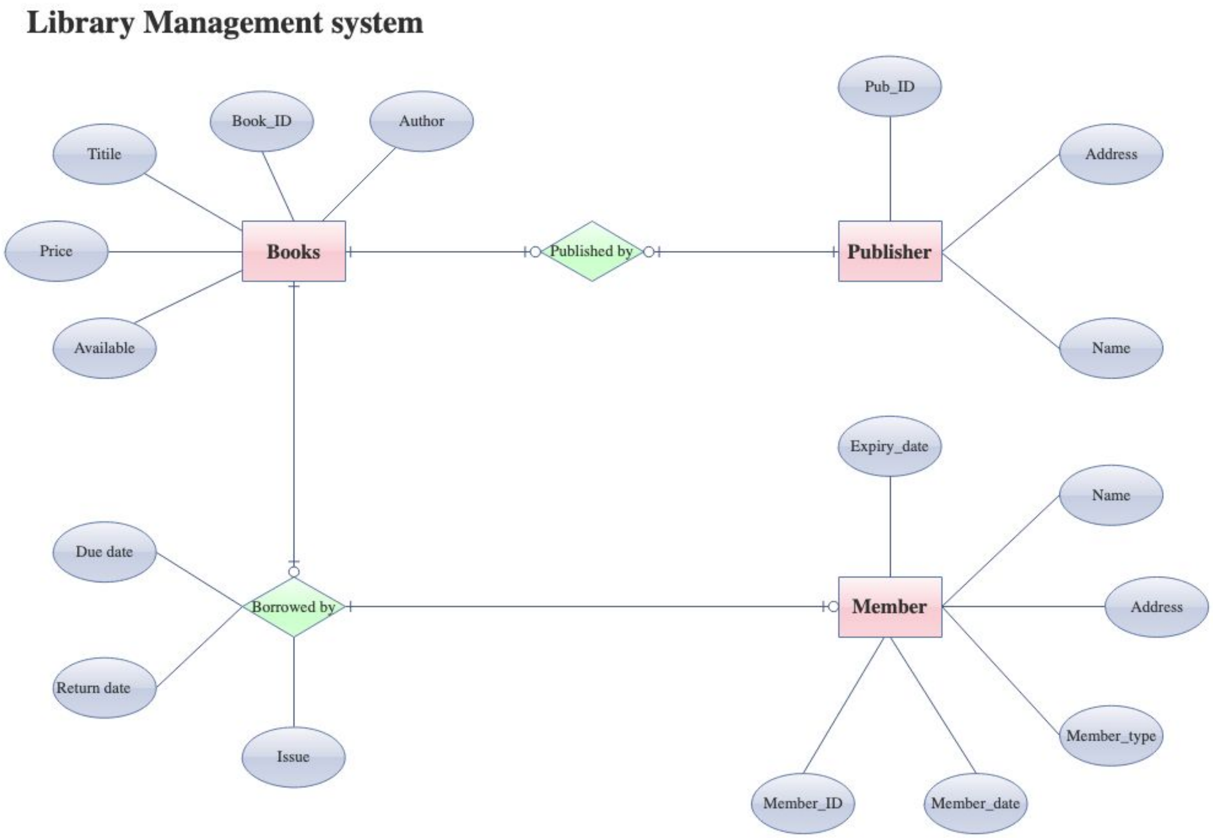


Different notations to represent cardinality





Example of a ER diagram for a Library Management System



Participation Constraints and Existence Dependencies

Participation CONSTRAINTS and CARDINALITIES must be ENFORCED when IMPLEMENTING the database

The participation constraint specifies whether the existence of an entity depends on its being related to another entity via the relationship type.
- specifies the minimum number of relationship instances that each entity can participate - minimum cardinality constraint.

Total Participation

• Each entity is involved in the relationship: represented by double or thicker lines

Partial participation

• Not all entities are involved in the relationship: represented by single lines

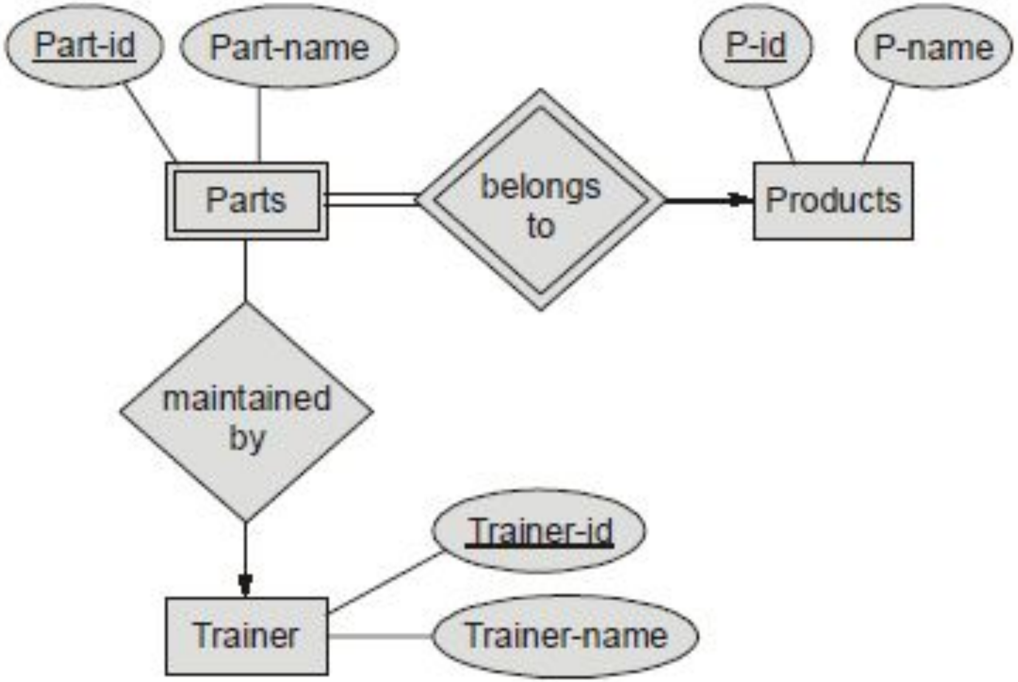
Strong and Weak Entity Types

- Entity types that do have a key attribute - are called <u>strong entity types</u>.
- Entity types that do not have key attributes of their own are called <u>weak entity types</u>.
- Weak entities are identified with one of their attribute values in combination with another entity type (owner entity type).
- <u>Weak entity type always has a total participation constraint with respect to its identifying relationship because a weak entity cannot be identified without an owner entity.</u>
- **Notation:** A weak entity type and its identifying relationship are surrounding their boxes and diamonds with double lines

Weak Relationships

- ❖ Weak relationship is the identifying relationship of a weak entity type:
  - connection that exist between a weak entity type and its owner

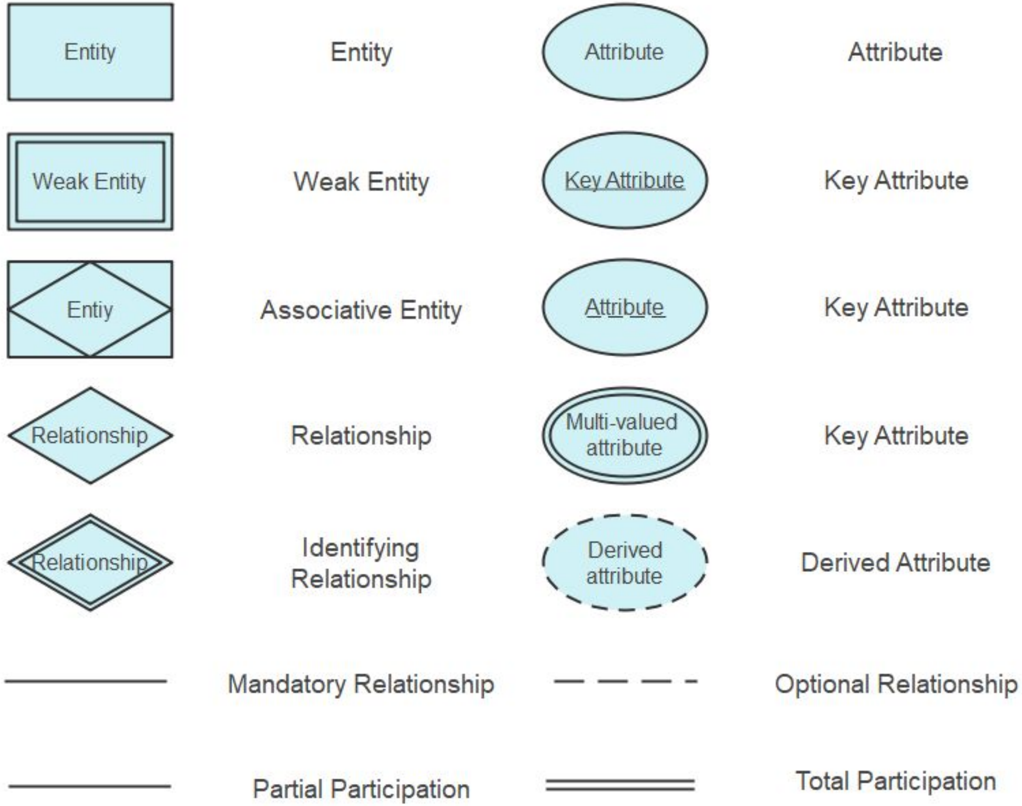
- ❖ A weak relationship is indicated by a doubly - outlined diamond



Design Choices

- It is occasionally difficult to decide whether a particular concept should be modeled as an entity type, an attribute, or a relationship type.
- The schema design process is an iterative refinement process, where an initial design is created and then iteratively refined until the most suitable design is reached.
- General Recommendations:
  - It is often the case that a pair of such attributes that are inverses of one another are refined into a binary relationship.
  - Once an attribute is replaced by a relationship, the attribute itself should be removed from the entity type to avoid duplication and redundancy.
  - An attribute that exists in several entity types may be elevated or promoted to an independent entity type.

Summary of the notation for ER diagrams




Q & A