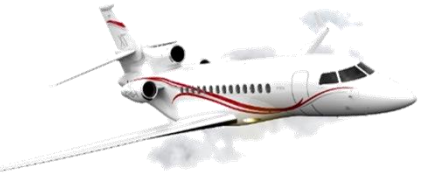
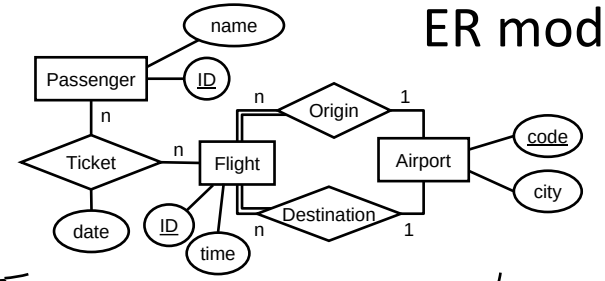
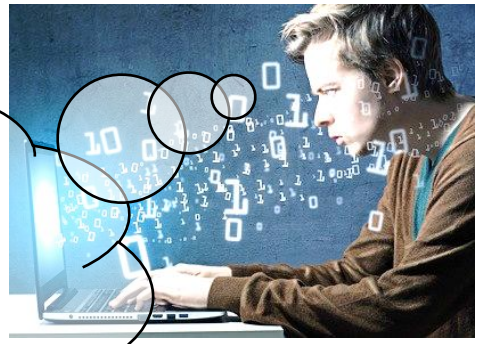
Contents



ER model





Flight

number	origin	destination	time
345	MAD	CDG	12:30
321	MAD	ORY	19:05
165	LHR	CDG	09:55
903	CDG	LHR	14:40
447	CDG	LHR	17:00

Passenger

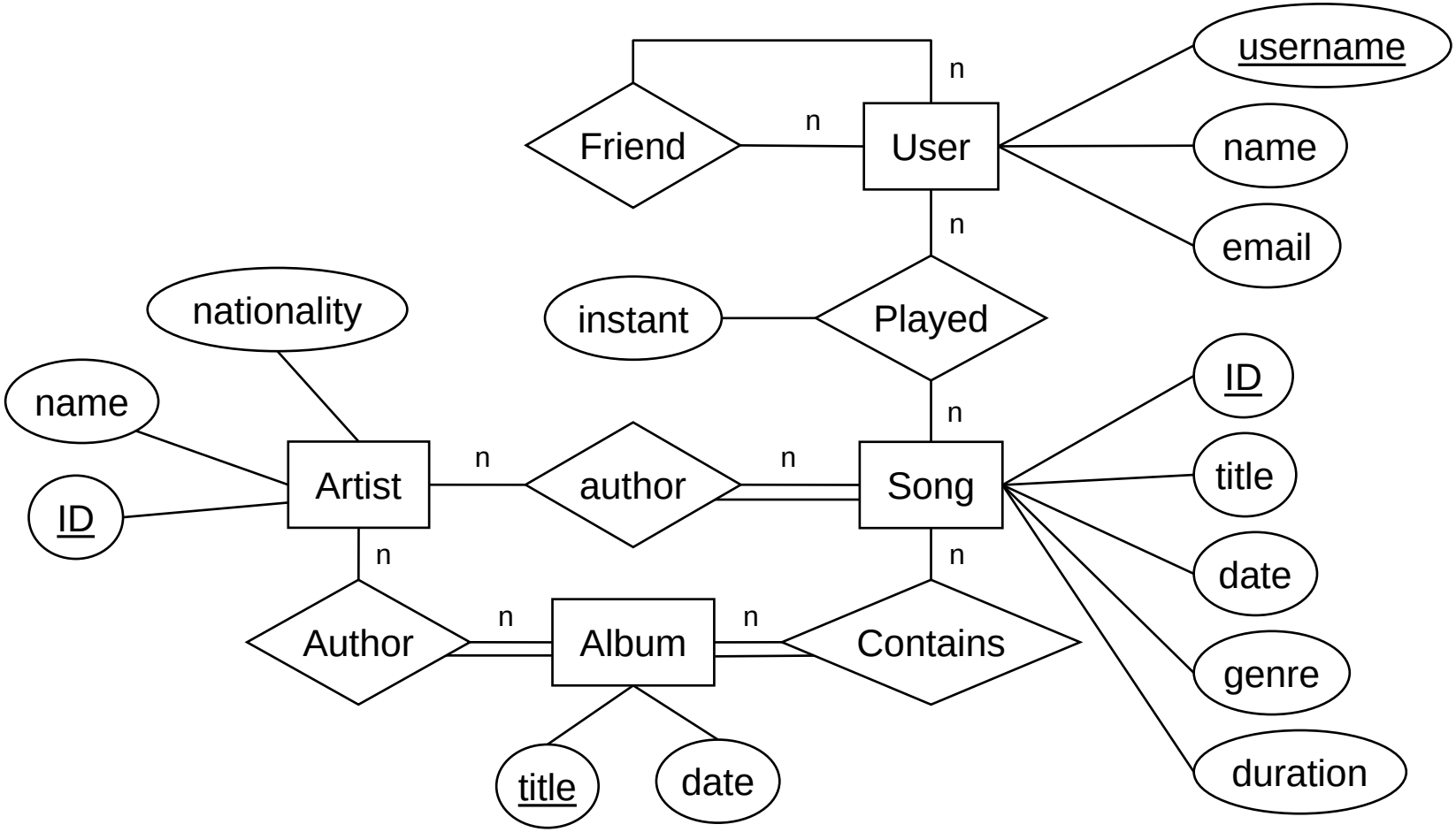
id	name
123	Mary
456	Peter
789	Isabel
321	Peter

Ticket

id	number	date	price
789	165	07-01-11	210
123	345	20-12-10	170
789	321	15-12-10	250
456	345	03-11-10	190

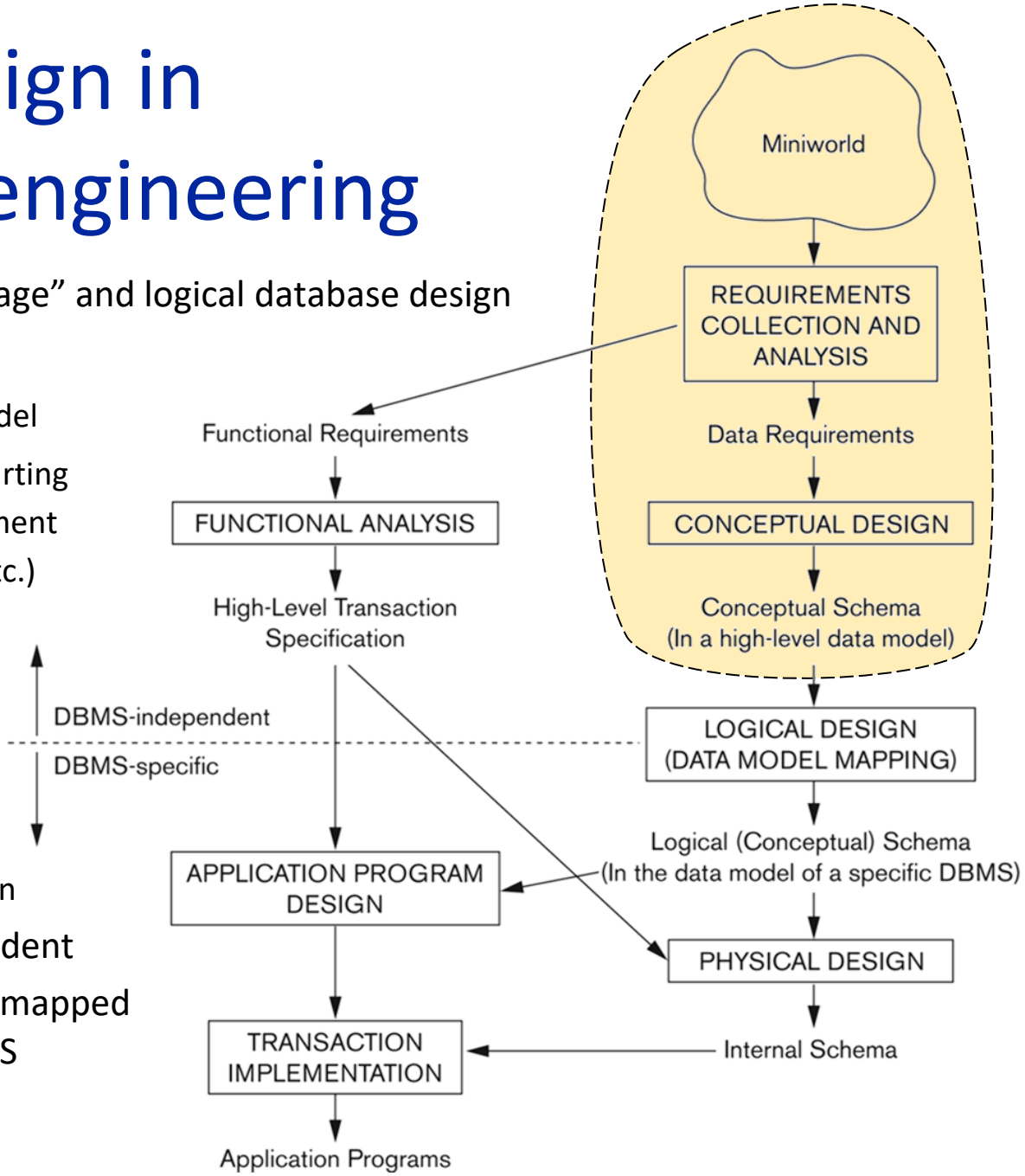
SQL

Entity-relationship model



ER design in application engineering

- Transition between “natural language” and logical database design
- Use in software engineering
  - Conceptual design of the data model
  - It is difficult to write SQL when starting from scratch (application requirement understanding, user interviews, etc.)
- More tractable “language” than SQL to capture and model data requirements
- Graphically visualizable
  - But with formally defined elements and alternative set notation
- Abstract: implementation-independent
- Once the ER model is defined, it is mapped to the relational model and a DBMS
  - Relational schemas, restrictions
  - DBMS: SQL, tables



Elements of the ER model

- Entity
- Entity type (concept)
- Attribute
- Attribute properties
- Relationship
- Key

Entity

- Entity
  - "Thing", name
  - Example: *La leyenda del tiempo*, *Pink Floyd*, *Mary*
- Entity type (concept)
  - "Complete" set of entities as to the common attributes

    { *Mary*, *John* }, { *The Beatles*, *José Monge Cruz*, *Pink Floyd* } → can be entity types

    { *The Beatles*, *La leyenda del tiempo* } → is not an entity type

    ∅ and the set of all entities → are entity types

    { *Mary*, *John*, *The Beatles*, *José Monge Cruz*, *Pink Floyd* } → can be an entity type?
  - Intension vs. extension

    Intension: Artist, User, Song, Album

    Extension: { *The Beatles*, *José Monge Cruz*, … } ≡ Artist

    { *Mary*, *John*, … } ≡ User

Attributes

- Attribute: function between an entity type and a domain

  nick : User $\rightarrow$ string name : Artist $\rightarrow$ string

  email : User $\rightarrow$ string date : Song $\rightarrow$ date

  name : User $\rightarrow$ string …

- Domain: set of valid values for an attribute
  - string, numeric, date, ciudad, zip code, etc.
- Attribute properties
  - Simple vs. composite

    address : Person $\rightarrow$ string $\times$ integer $\times$ city $\times$ … (‘c/ Mayor’, 15, ‘Madrid’, …)

    May have nested levels
  - Single-valued vs. multivalued

    telephone_number : Person $\rightarrow \mathcal{P}$(string) { ‘911234567’, ‘612345678’ }
  - Derived: e.g. age, number of friends
- Valor NULL
  - Any attribute can take this value
  - Non-existing value (e.g. an address that does not have a floor) or unknown value (e.g. someone’s birth date), or we do not know if a value exists (e.g. passport number)

Keys

- Superkey
  - Set of attributes whose combination is unique for an entity type
  - For instance, the set of all attributes of an entity type is a superkey (trivial)
  - Examples: username + name is a superkey of User
    Is ID a superkey of Person?
- Key
  - A minimal superkey, a.k.a. candidate key
  - Equivalent to UNIQUE in SQL
  - Examples: nick + name is not a key for User
    nick is a key
    email is a key
- Primary key
  - A key that is designated as primary for an entity type
  - It plays a specific role in indexing tables (we shall see later on…)
  - Equivalent to PRIMARY KEY in SQL
  - The choice between candidate keys is arbitrary
  - Graphic notation: underlined

Relationships

- Relationship = subset of the Cartesian product of several entity types

  Author $\subset$ Song $\times$ Artist

  Contact $\subset$ User $\times$ User

  Author = { (*Norwegian wood*, *The Beatles*), (*The Wall*, *Pink Floyd*), … }
- Relationship type vs. instance
  - Relationship type = set of relationship instances
- Cardinality
  - 1-1, 1-*n*, *n*-1, *n*-*n*, ranges for the '*n*'
  - Partial participation (cardinality ≥ 0) vs. total participation (cardinality > 0)
  - Examples?
- Degree
  - Binary: the most usual
  - N-ary: Ambassador $\subset$ Person $\times$ País $\times$ País

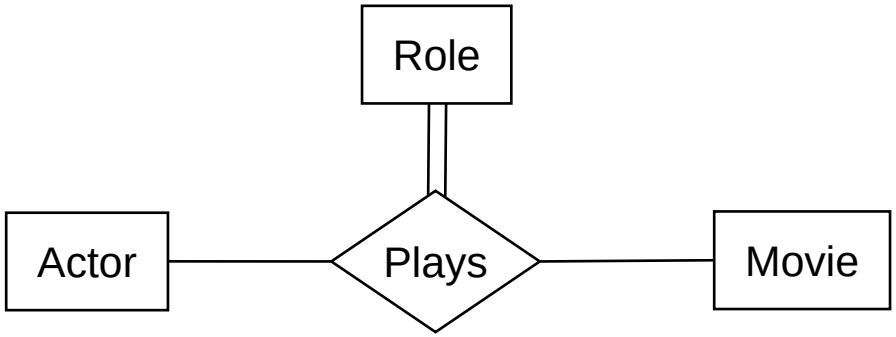
    Plays $\subset$ Actor $\times$ Movie $\times$ Role

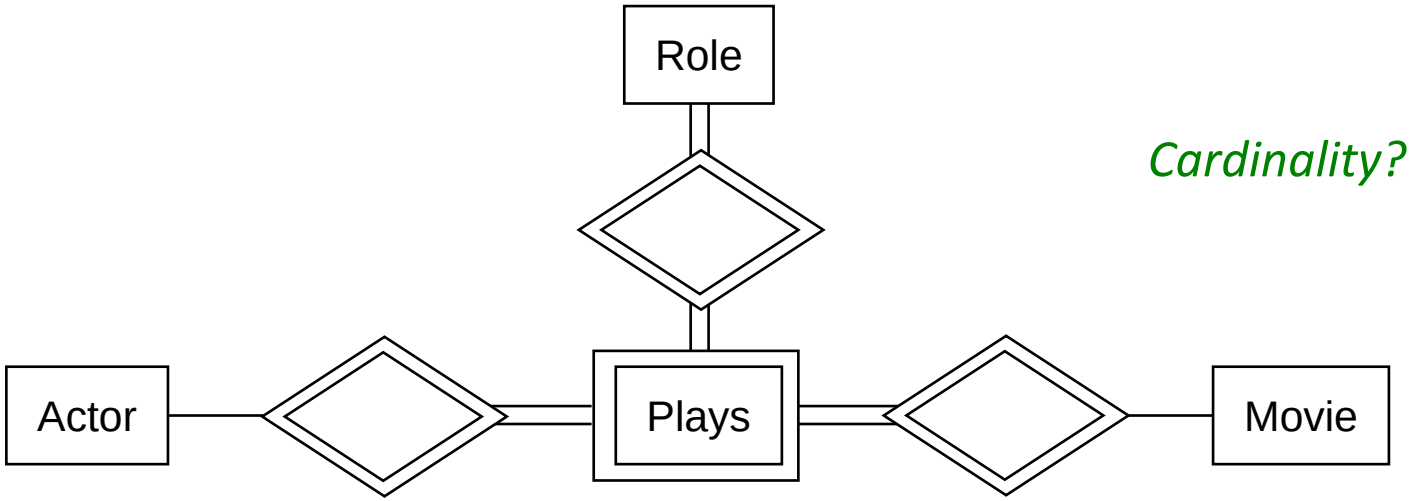
    Lineup $\subset$ Team $\times$ Player $\times$ Match

    Change $\subset$ Player $\times$ Player $\times$ Match

    They are equivalent to an entity with as many relationships as the degree

N-ary relationship vs. entity with n binary relationships





Cardinality?

Relations (cont)

- Relation attributes

  Ticket $\subset$ Passenger $\times$ Flight

  Date : Ticket $\rightarrow$ date

  Seat : Ticket $\rightarrow$ string

  Price : Ticket $\rightarrow$ float

  Lineup $\subset$ Team $\times$ Player $\times$ Match

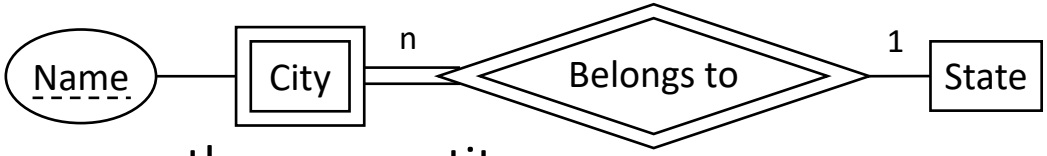
  Position: Lineup $\rightarrow$ Integer $\times$ String

  Position (*Chelsea*, *Pedro Neto*, *Premier league match week 5*) = (10, 'forward')

- Roles
  - Optional tags, clarify the semantics
  - May distinguish direction of the relationship
  - Especially useful in asymmetric reflexive relationships

Weak entities

- Lack a key
- Depend on a relation with another entity in order to be identifiable
  - They have a total participation in this relationship
  - The relationship is 1-1 or 1-*n* (*n* on the side of the weak entity)
  - They have no meaning without the entity they depend on
- They usually have a partial key
  - It distinguishes from each other the weak entities that depend on the same entity
  - The key of the identifying entity plus the partial key makes up a full key for the weak entity
  - If the relationship is 1-1 no partial key is needed
- Example
  - Album tracks (partial key: track nr.)
  - Cities in a USA state (partial key: city name)



- Can depend on more than one entity
  - Example: weak entity “Cast” relating Movie, Actor, Role

Summary of graphic notation