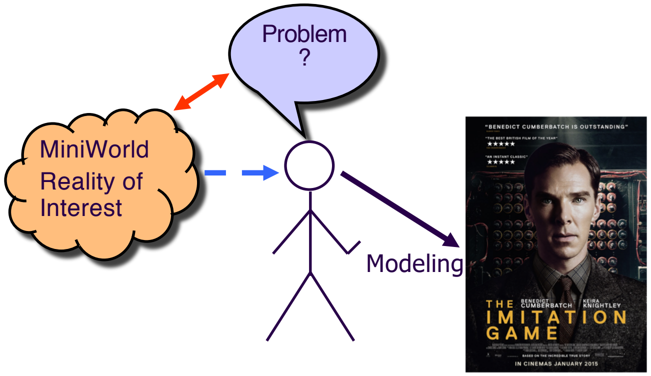
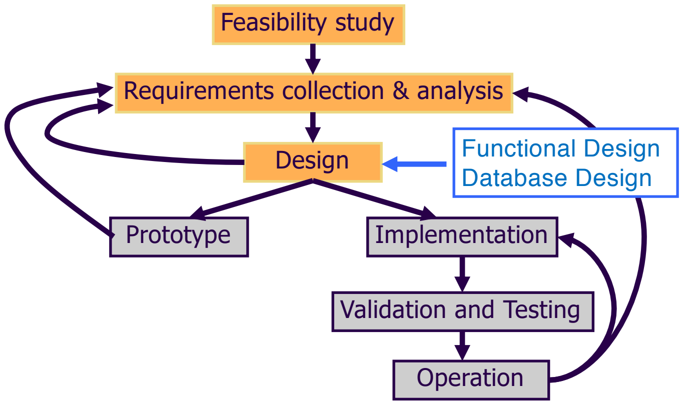
Conceptual Database Design & ER-Model

- ER-Model
- ER-Diagrams
- EER Model & Diagrams

Database System Design/Data Modeling



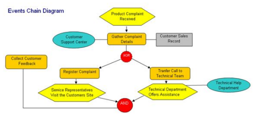
Database System Life Cycle

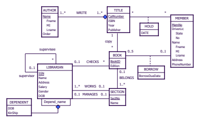


Functional Design

- High-level specification of Transactions
  - DBMS-independent
  - Event diagrams, UML

Application, Business Logic





- Application program design
  - DBMS-specific (db Schema together with DML)
  - Language and environment-specific

Database Design

- Database design is the activity of specifying the schema of a database in a given data model

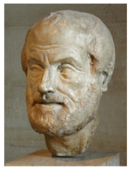
- Three categories:
  - Conceptual database design
  - Logical database design
  - Physical database design

Database Design

- Conceptual database design
  - An abstract but complete description of the DB
  - Implementation independent (*semantic clarity*)
  - E.g., conceptual model: E-R Model, UML
- Logical database design
  - The conceptual database schema
  - Formal schema in an *implementation* data model
  - E.g., Relational, O-O, O-R, Network, hierarchical
- Physical database design
  - Internal schema: Internal storage organization of objects, implementing the conceptual model

Aristotle (Greek: Ἀριστοτέλης Aristotélēs)

384 BC – 322 BC



- The first to create a comprehensive system of philosophy, encompassing morality and aesthetics, logic and science, politics and metaphysics.

- Taxonomy [Physica: physical sciences]
  - living things
  - their relationships
  - prototype or exemplar

Entity-Relationship Model (P. Chen, 1976)

Two Semantics primitives

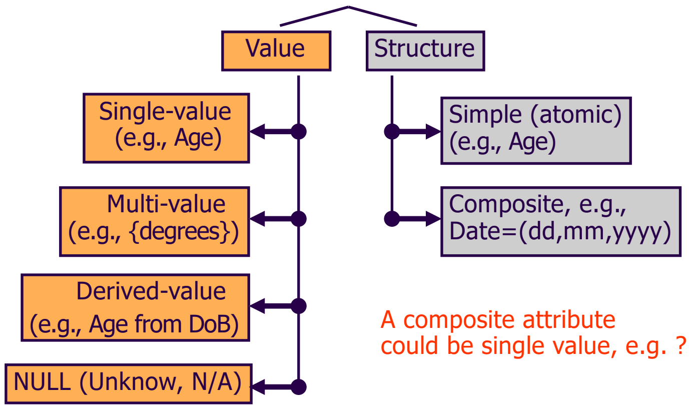
- Entities
  - Objects with physical existence, e.g., Peter, Mary, Peter' s house, etc.
  - Objects with conceptual existence, e.g., University, Course, Account, etc.

- Relationships
  - Associations between two or more entities e.g., Peter *married* Mary, Mary *studies* Physics, etc.

Attributes

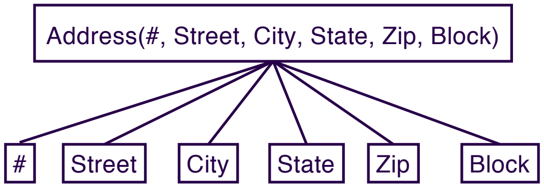
- Entities are characterized by their attributes
  - Peter has an age,
  - Mary’s car has a color
- Relationships may also have attributes
  - Peter married Mary on Jan 7

Attribute Classification

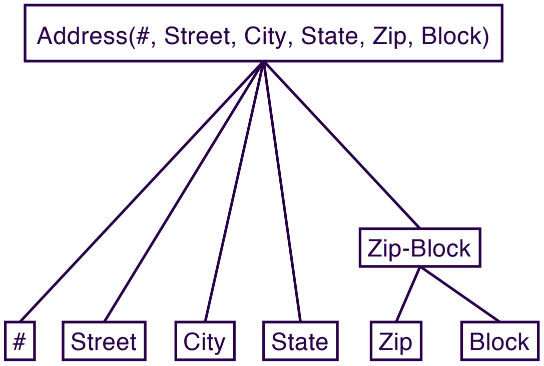

Value
Structure
Single-value (e.g., Age)
Multi-value (e.g., {degrees})
Derived-value (e.g., Age from DoB)
NULL (Unknow, N/A)
Simple (atomic) (e.g., Age)
Composite, e.g., Date=(dd,mm,yyyy)

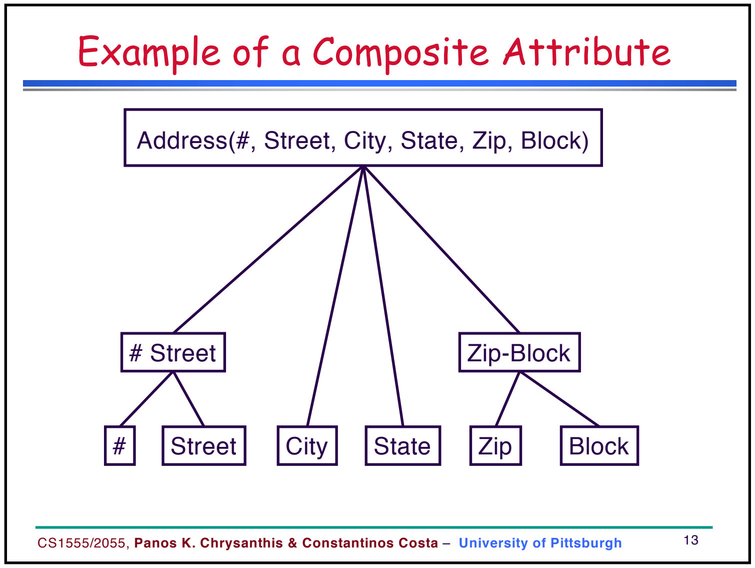
A composite attribute could be single value, e.g. ?

Example of a Composite Attribute



Example of a Composite Attribute





Example of a Composite Attribute

Address(#, Street, City, State, Zip, Block)

# Street

Zip-Block

#

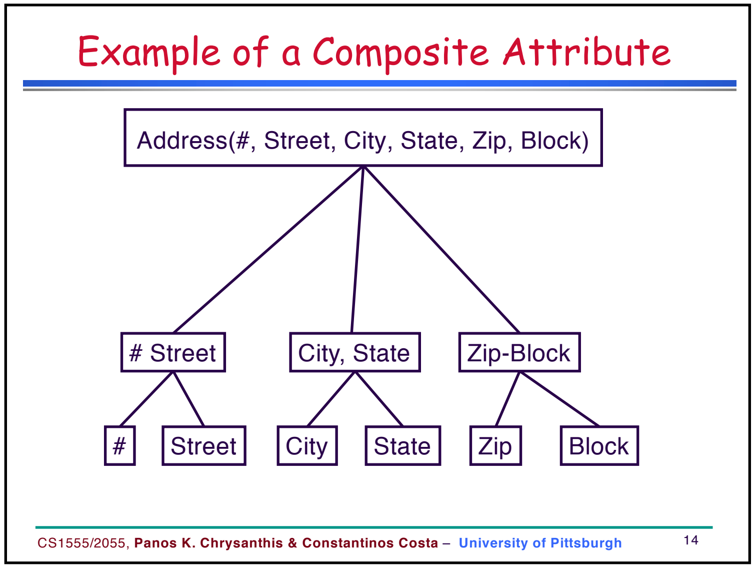
Street

City

State

Zip

Block



Example of a Composite Attribute

Address(#, Street, City, State, Zip, Block)

# Street

City, State

Zip-Block

#

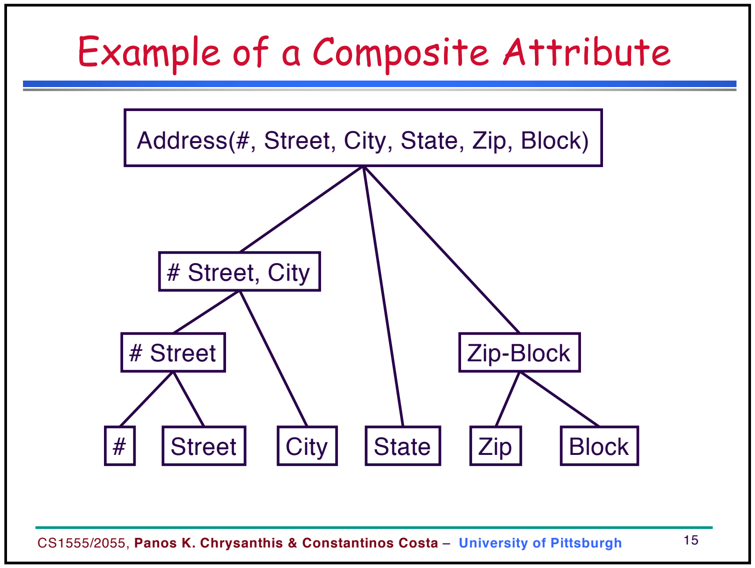
Street

City

State

Zip

Block



Example of a Composite Attribute

Address(#, Street, City, State, Zip, Block)

# Street, City

# Street

Zip-Block

#

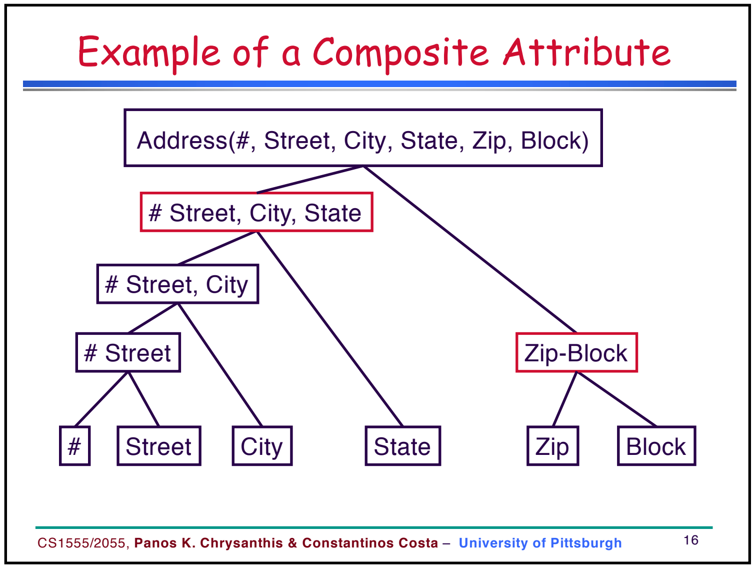
Street

City

State

Zip

Block



Example of a Composite Attribute

Address(#, Street, City, State, Zip, Block)

# Street, City, State

# Street, City

# Street

Zip-Block

#

Street

City

State

Zip

Block

Entity Types

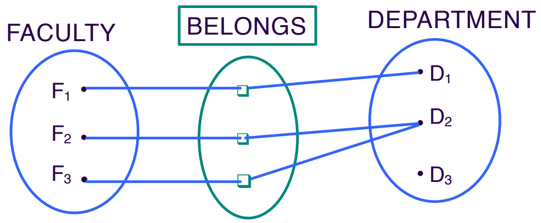
- All similar (same attributes) entities are grouped into sets, an entity type
- Entity type schema specifies the common structure:
  - type name
  - entity attributes (Domain, value set)
  - constraints on entities
- E.g.,
  FACULTY: Name(FN,LN,MI), DoB, SSN, {Degree}, Rank
  - FN:String(15), LN: String(15), SSN: String(9), etc.
  - DoB: DD/MM/YYYY
  - Degree: {BS,MS,PhD}
  - Rank: {Lecturer, Assistant, Associate, Full}

Uniqueness or Key Constraint

- Entities are distinguished by using various keys
- A key is a uniqueness constraint on attributes
- A Key is defined over one or more attributes
  - *SSN, StudentID, Car License Plate: State and Number*
- Superkey: Any combination of attributes that uniquely identifies an entity
  - *Name and SSN, Name and StudentID*
- Candidate Key is a minimal superkey
  - *E.g., SSN and StudentID*
- Primary Key is one of the candidate keys (SSN)
- Alternative keys are the remaining candidate keys
  - *Primary key is underlined, alternative are over-lined*

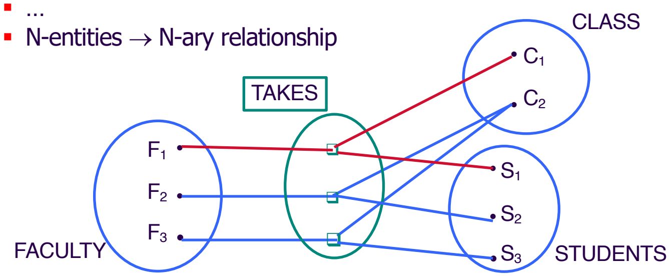
Relationship Types

- Relationship Types: sets of relationships that are homogeneous in participating entities
  - BELONG:<FACULTY, DEPARTMENT>
  - ENROLLS:<STUDENT, SECTION>



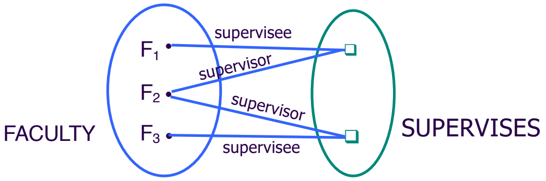
Degree of a relationship

- Degree of a relationship is the number of participating entity types:
  - 2-entities → binary relationship
  - 3-entities → ternary relationship
    - TAKES:<STUDENT, CLASS, FACULTY>
  - ...
  - N-entities → N-ary relationship



Degree of a relationship

- Recursive relationships that involve more than once the same entity type with different Roles:
  - SUPERVISES:<supervisor-faculty, supervisee-faculty>



Constraints on Relationship Types

- Cardinality ration: Specifies the number of relationship instances that an entity can participate in.
  - **1:1** Departments having Chairpersons
  - **N:1** Children having Mothers
  - **1:N** Mothers having children (inverse of N:1)
  - **M:N** Students enrolling in Class Sections
- Participation:
  - **Total** → Existence of entity depends on the existence of a related entity. E.g., Classes have total participation to OFFER_BY dept.
  - **Partial** → Some entities are not related to other entities. E.g., Faculty have partial participation to CHAIR of a dept.

Strong and Weak Entities

- Strong or ordinary Entities:
  - Have independent existence in the mini-world
  - They are part of the care of the application
- Weak Entities:
  - They are dependent on another entity
  - Identify owner is the specific entity on which the weak entity depends
  - No key attribute; are distinguishable through an identifying relationship and a discriminator or partial key
  - Identifying relationship is always total participation
  - It may be represented as multi-value, composite attribute of owner (When isn' t this possible?)