Using Entity-Relationship Diagrams To Count Data Functions

Ian Brown, CFPS
Booz Allen Hamilton
8283 Greensboro Dr.
McLean, VA 22102
USA

Booz | Allen | Hamilton

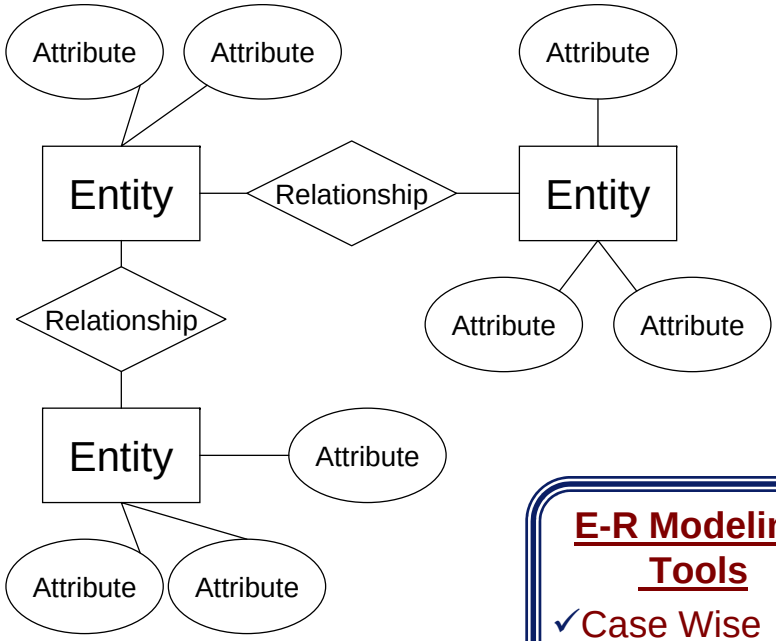
Contents

- What Is an Entity-Relationship (E-R) Diagram?
- E-R Vocabulary
- How to Read E-R Diagrams
- Applying *IFPUG Counting Practice Manual (CPM) 4.2* Rules to E-R Diagrams

This document contains material which has been extracted from the IFPUG Practical Guidelines for Counting Logical Files. It has been reproduced in this document with permission of IFPUG & NESMA

An E-R diagram is a set of common constructs and conventions used to create a model of users' data

- E-R diagrams also known as **logical** data models (LDM)
- Data modeling is the process of creating a **logical** representation of the structure of a database
  - To be accurate, the model must support the **users' view** of the data
  - Typically completed in design phase of a development effort
- Common notations
  - Chen model
  - Barker's notation
  - Information engineering (crow's feet)
  - IDEF1X
  - UML



E-R Modeling Tools

- ✓Case Wise
- ✓ERwin
- ✓S-Designer
- ✓ER/Studio
- ✓Visio

There is no single, generally accepted standard E-R model…

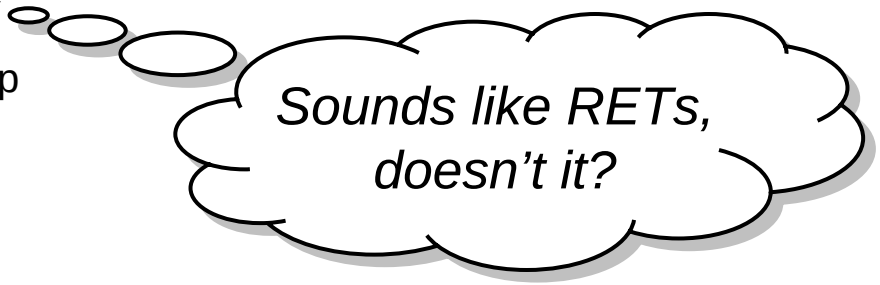
…But a common terminology provides some consistency between E-R variations

- **Entity:** represents a set of real or abstract objects about which a system manages and maintains information
  - **Independent Entity:** can be uniquely identified without determining their relationship to another entity
  - **Dependent Entity:** existence of an instance (record) relies upon its relationship to another entity
    - **Key:** a field that is a unique identifier for each data entry
    - **Associative Entity:** further describes the relationship between two other entities; depends upon two or more “parent” entities and takes the entire key from both parent entities
    - **Attributive Entity:** further describes one or more characteristics of another entity
    - **Subtype Entity:** augments a unique occurrence of an entity with additional characteristics and/or relationships
- **Attribute:** represents a field, or data element, of a given entity
- **Cardinality:** the maximum or minimum number of elements allowed on each side of the relationship; specifies how many instances of one entity relate to one instance of another entity

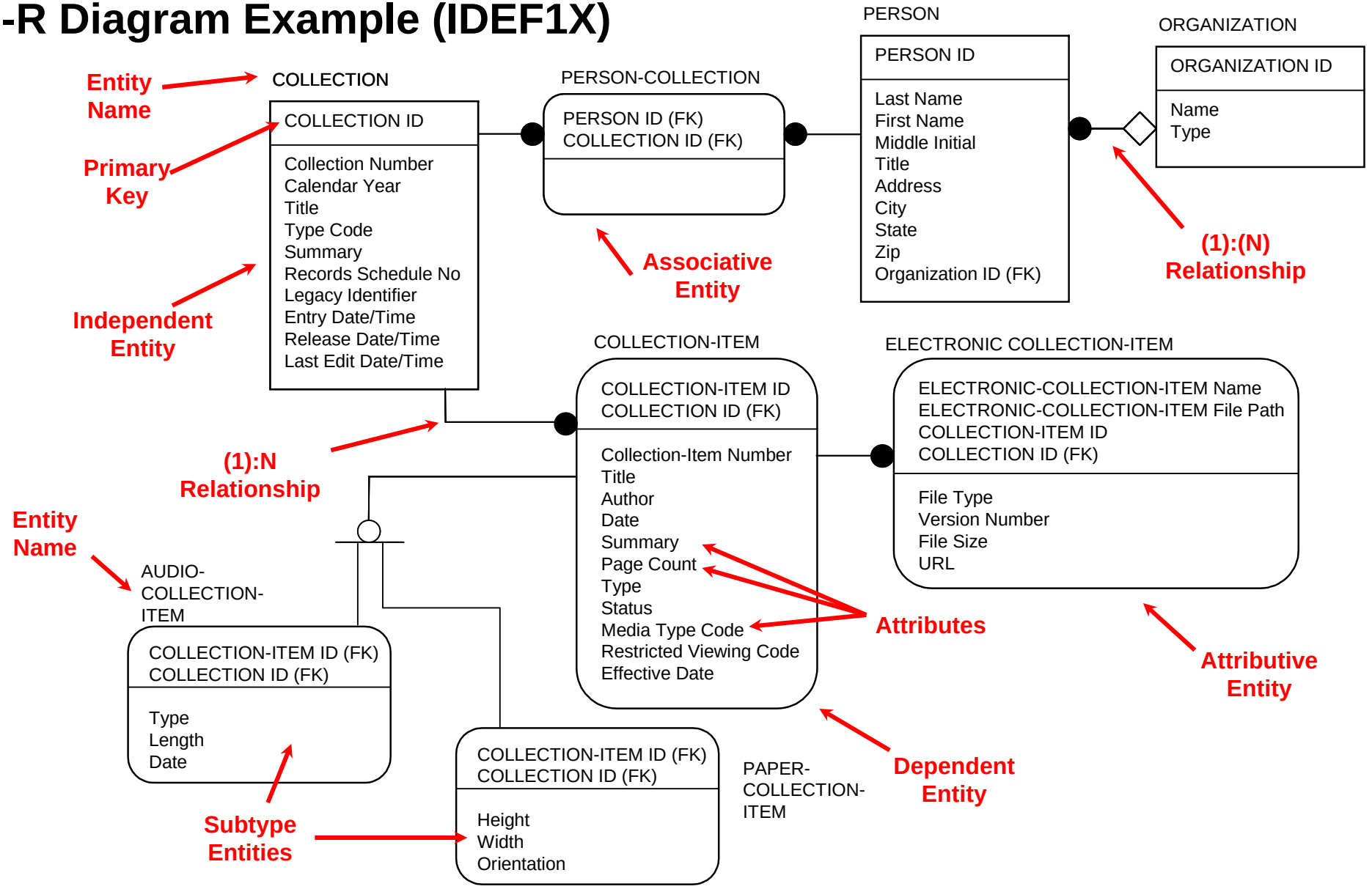
Relationships define the nature of the connection between attributes within entities

- One-to-one [1:1] – A determines B and B determines A; two attributes functionally determine each other
  - Example: person name and driver license number
- One-to-many [1:N] – A determines B, but B does not determine A; single instance related to multiple instances
  - Example: CD title and artist name
- Many-to-many [N:M] – A does not determine B and B does not determine A
  - Example: student name and course number

- Relationships can also be *optional* or *mandatory*
  - Parentheses used to indicate optional relationship
  - (1):N or 1:(N)



E-R Diagram Example (IDEF1X)



Applying *CPM 4.2* Rules to E-R Diagrams

The basic methodology follows CPM 4.2 guidance

① Identify and classify logical files

② Identify record element types (RETs)

③ Identify data element types (DETs)

A logical file is a logical group of data from the user's perspective

① Identify and classify logical files

- Logical files can consist of one or more data entities
- Basic identification and classification process

① **Identify entities that should be considered for counting**

② **Identify the user/business view of the data**

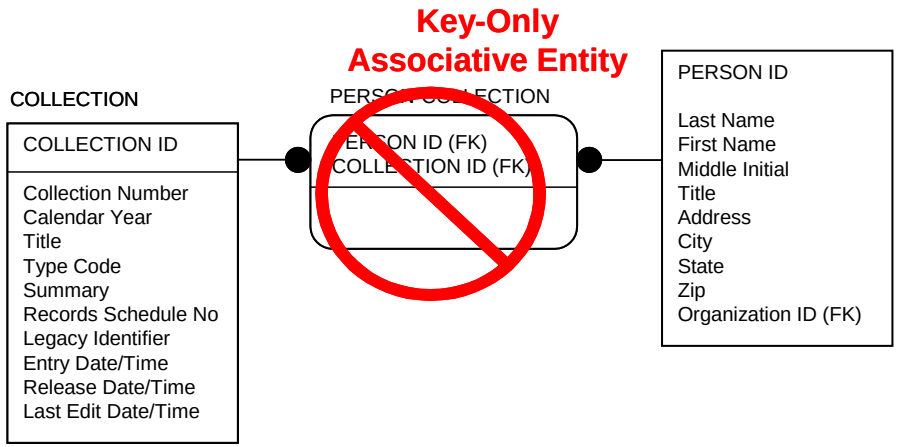
③ **Classify identified logical files as ILF or EIF**

Not all entities in an E-R diagram should be included in a function point analysis

① Identify and classify logical files

① Identify entities that should be considered for counting

- Clear guiding principle: do not consider entities that are not significant to and required by the end user
- Do **NOT** include
  - Entities that contain technical attributes only (index files)
  - Associative entities that contain only technical attributes or are a result of design or implementation considerations
  - Associative entities that only have keys as their data elements
  - Entities that are not maintained by an elementary process, either within the boundary of the application or by another application (reference tables)



- Make sure to include logical files that exist based on user need, but may not show up on the E-R diagram, such as historical files

The users' business view of the data can be determined by how transactions will access that data

① Identify and classify logical files

② Identify the user/business view of the data

- Elementary process method
  - With guidance from analysts or end users, identify how *elementary processes* will use *external inputs* to maintain the entities
  - Entities created and deleted together strongly suggests a single file
  - Be wary of elementary processes that modify the data; modification transactions often refer to a single entity with a logical group

- Entity dependency method
  - Assess the independence or dependence of entities to appropriately group logically related data

Independent entity: meaningful and significant to the end user without presence of other entities

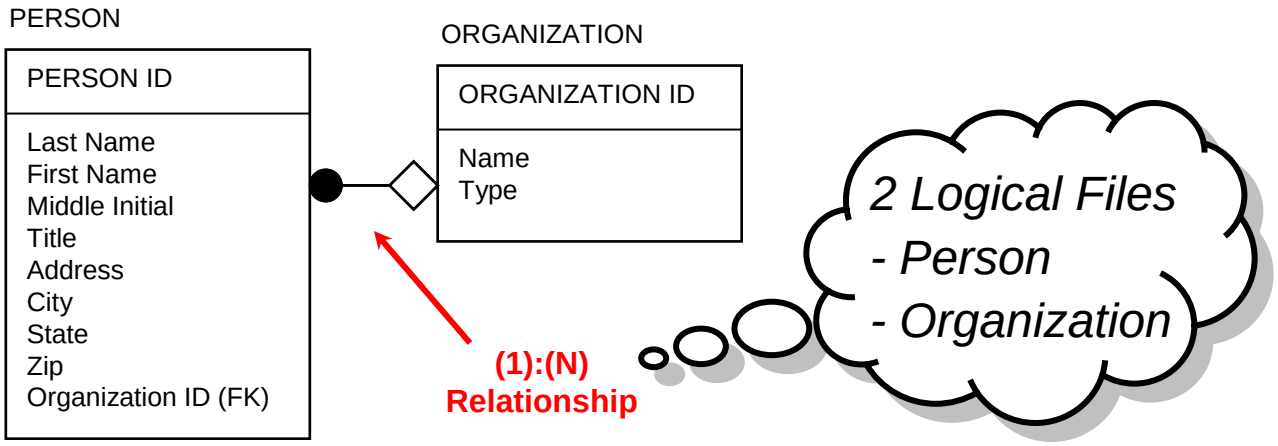
Dependent entity: has no significance to the business without presence of other entities

Entity (in-)dependence in a (1):(N) relationship

① Identify and classify logical files

② Identify the user/business view of the data

- Relationship is completely optional in both directions
  - Entities can exist completely independently
  - ***Count as two separate logical files***

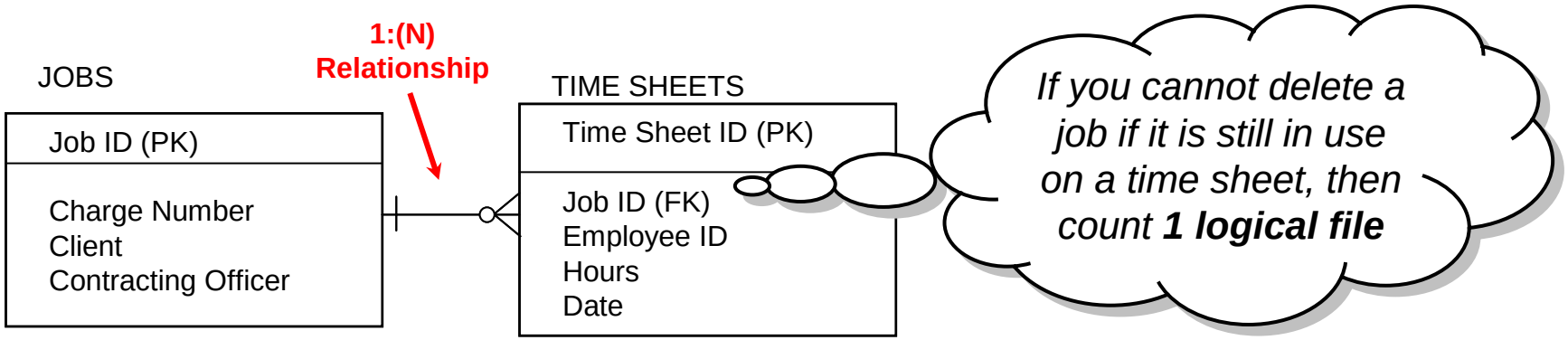


① Identify and classify logical files

② Identify the user/business view of the data

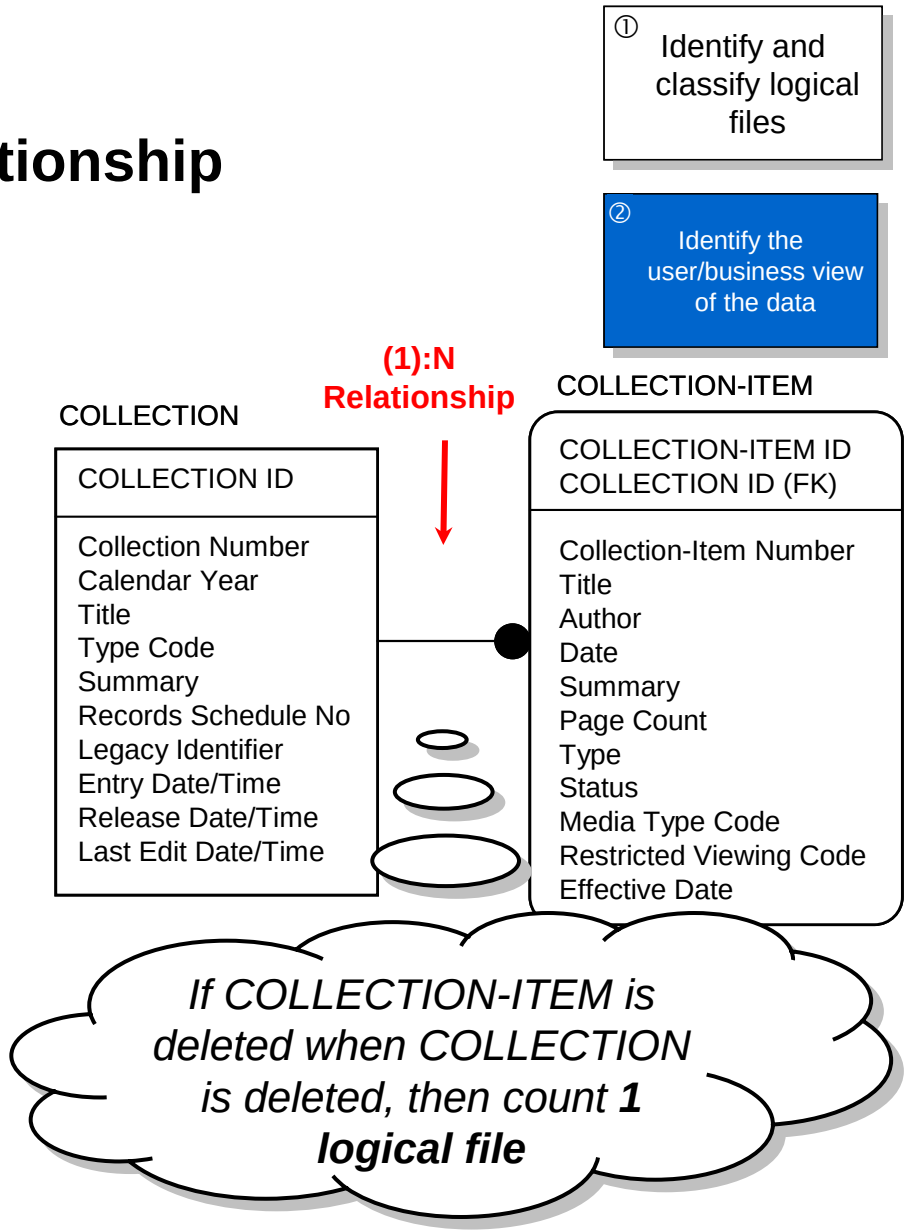
Entity (in-)dependence in a 1:(N) relationship

- Relationship is optional in one direction
  - An instance of entity A may exist to which none, one, or many instances of B are linked
- Ask questions
  - *Is B significant to the business apart from the A linked to it?*
  - *If we want to delete an instance of A, what happens to the linked instances of B?*
    - If all Bs are deleted with A, then B is dependent on A ***Count as a single logical file***
    - If A is not allowed to be deleted because Bs are still linked to it, then B is independent of A ***Count as two separate logical files***



Entity (in-)dependence in a (1):N relationship

- Relationship is optional in one direction (not too common)
  - Each instance of entity A must be assigned to one or many Bs, but B may or may not be assigned to an instance of A
- Ask questions
  - *Is A significant to the business apart from the B linked to it?*
  - *If we want to delete an instance of B, what happens to the linked instances of A?*
    - If all As are deleted with B, then A is dependent on B ***Count as a single logical file***
    - If B is not allowed to be deleted because As are still linked to it, then A is independent of B ***Count as two separate logical files***

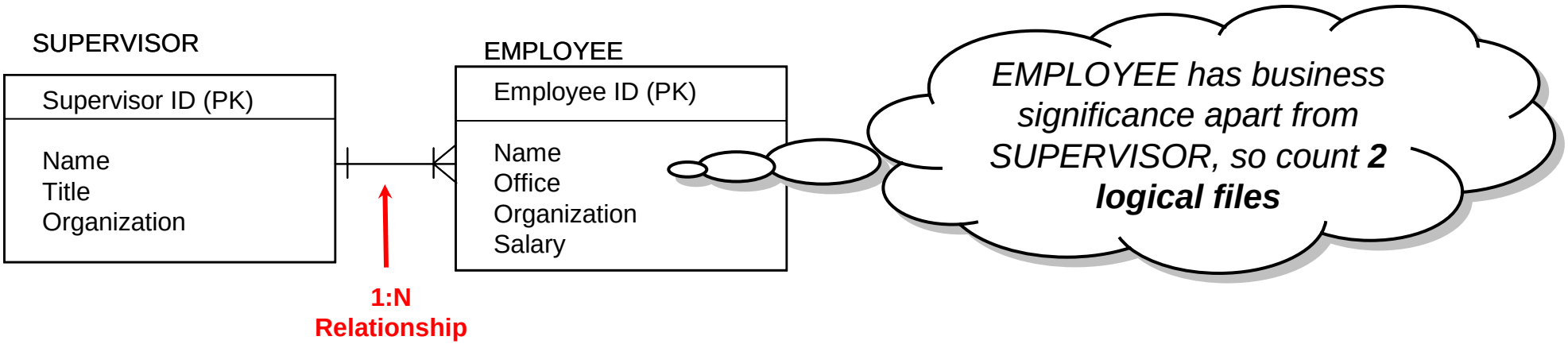


Entity (in-)dependence in a 1:N relationship

① Identify and classify logical files

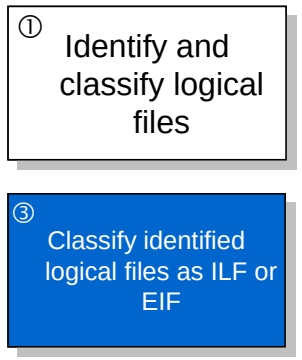
② Identify the user/business view of the data

- Relationship is mandatory in both directions
  - Each instance of entity A must be assigned to one or many Bs, and B must be assigned to an instance of A
- Ask questions
  - *Is B significant to the business apart from the A linked to it?*
  - If the answer is “no” ***Count as a single logical file***
  - If the answer is “yes” ***Count as two separate logical files***



Booz | Allen | Hamilton

Check identified logical files against ILF/EIF counting rules in *CPM 4.2*



- If the file is maintained by elementary processes within the boundary of the application being counted, classify as an internal logical file (ILF)

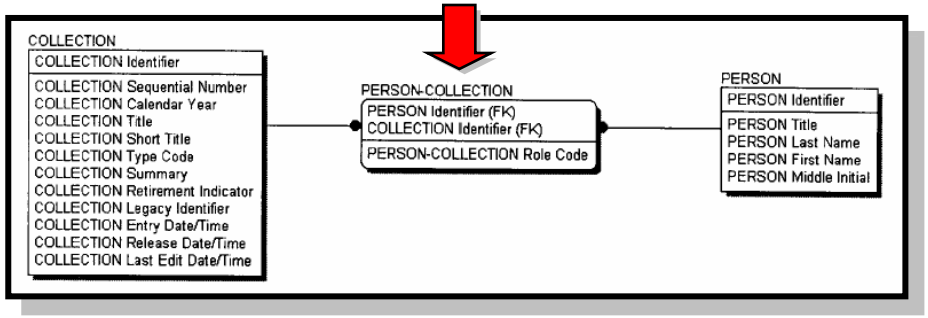
- If the file is only referenced by the application being counted, and it is maintained by an elementary process of another application, classify as an external interface file (EIF)

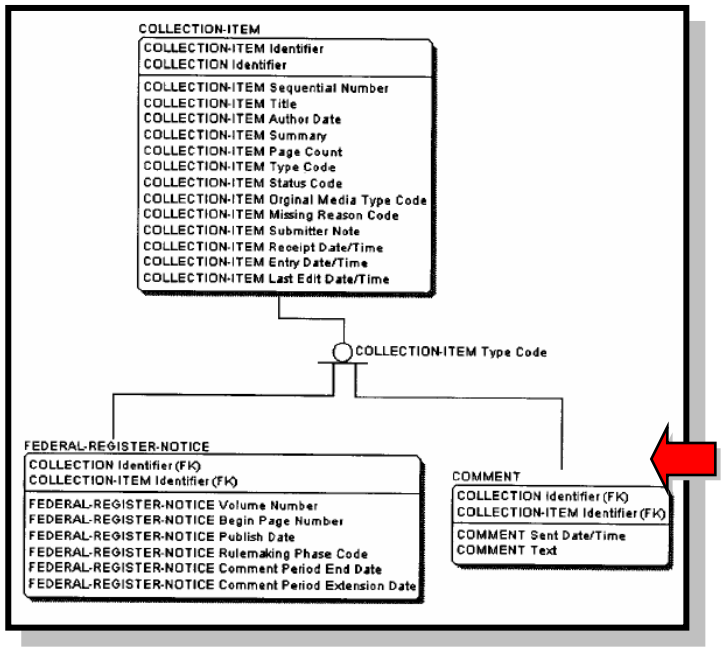
RETs represent the user's perspective on coherent subgroups of data within a logical file

② Identify record element types (RETs)

- This step in the methodology primarily looks at dependent entities in the E-R diagram that were determined not to be separate logical files

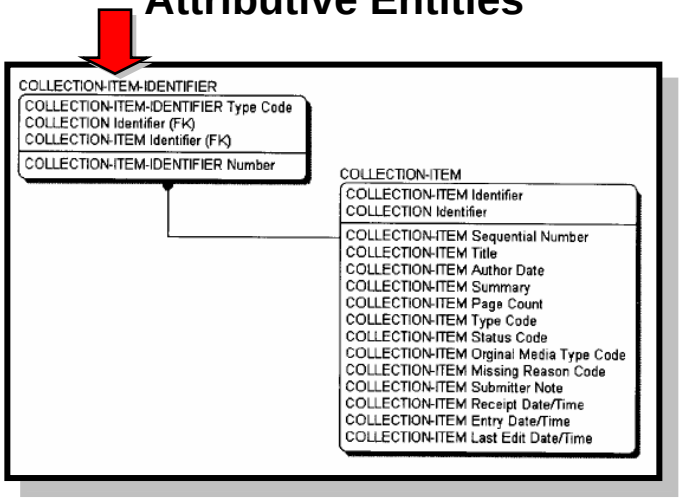
Associative Entities





Subtype Entities

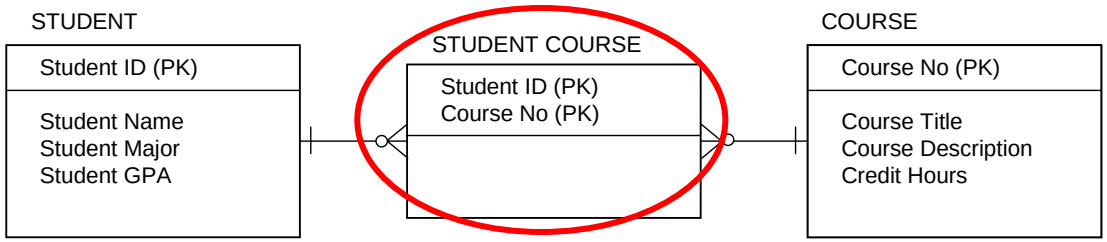
Attributive Entities



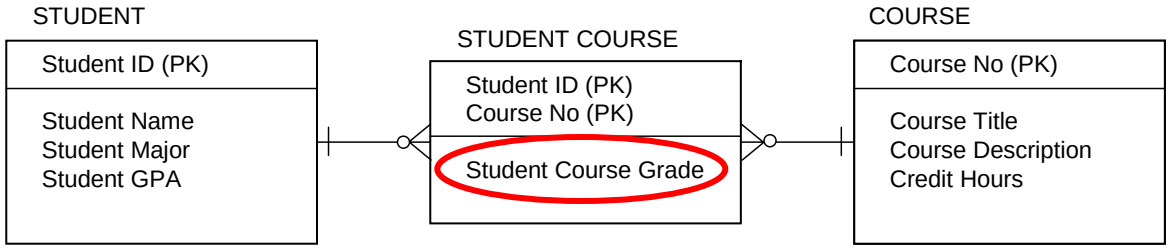
Associative entities help to define many-to-many relationships

② Identify record element types (RETs)

- In order to identify RETs, follow *CPM 4.2* guidelines
  - Identify the business need and use for the data in the associative entity
  - Review the nature of the data elements within the associative entity
- If the associative entity only contains primary keys from intersecting entities, then ***do not*** consider a RET



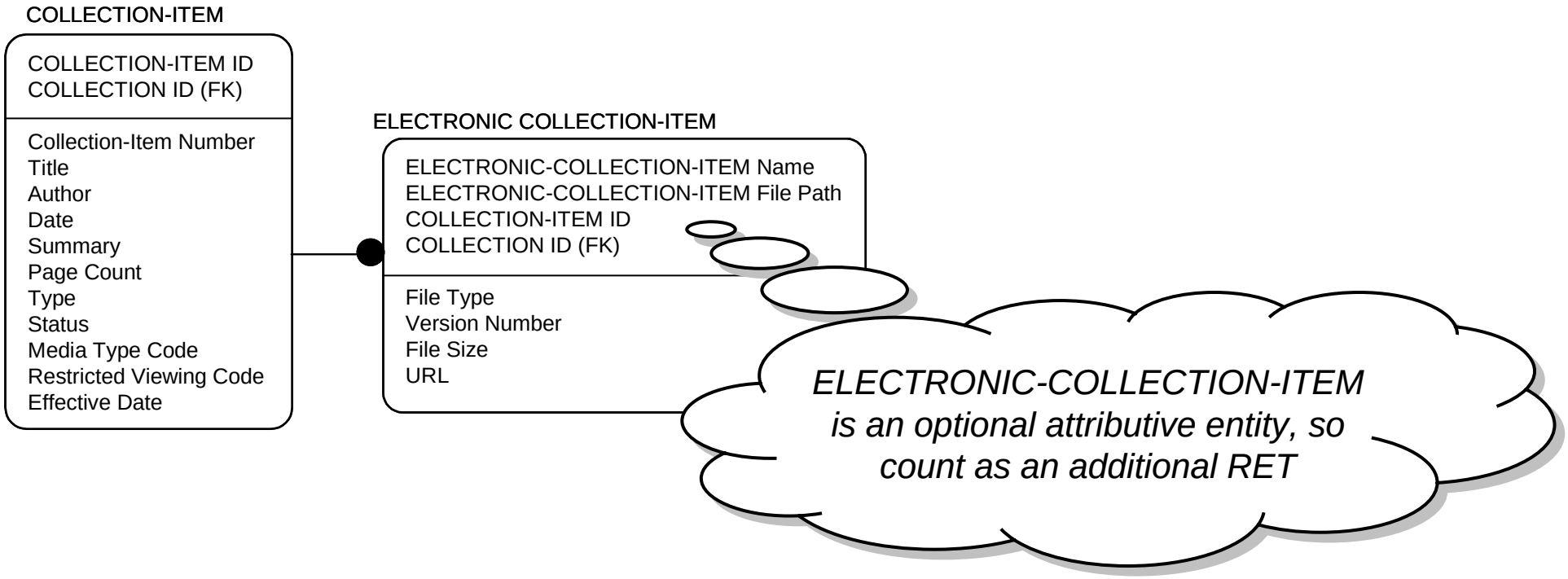
- If the associative entity contains at least one other attribute recognizable by the user, count as a RET



Attributive entities further describe one or more characteristics of another entity

② Identify record element types (RETs)

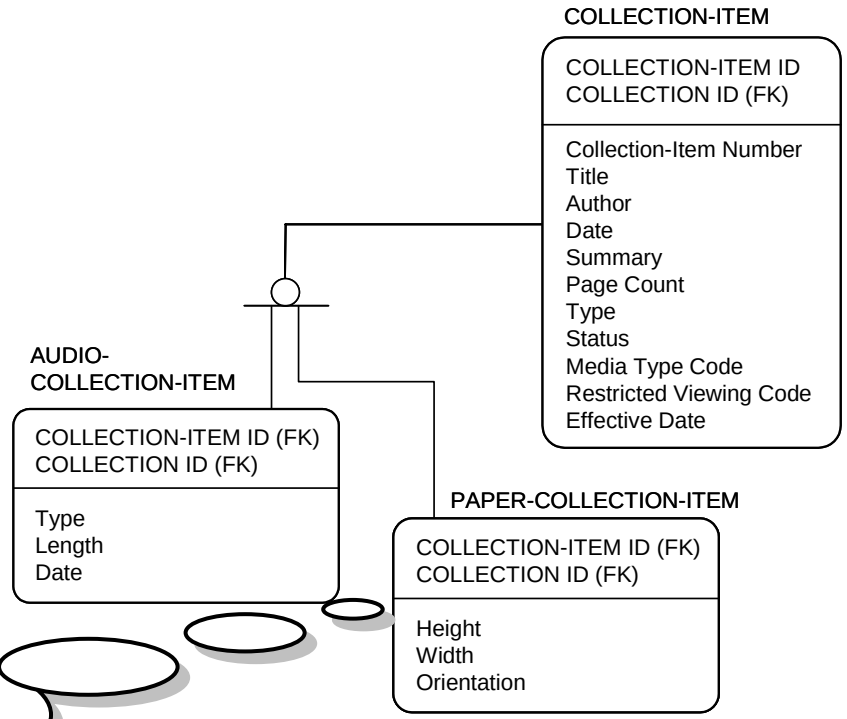
- The nature of attributive entities mean that it must be included in the function point analysis, either as a RET of a logical file or an extension of a logical file
  - Optional attributive entities are counted as RETS
  - Mandatory attributive entities are considered part of the larger logical file to which it is related



Subtype entities augment a unique occurrence of an entity with additional characteristics and/or relationships

② Identify record element types (RETs)

- Subtypes carry the entire key of its supertype
- Subtypes inherit the attributes and relationships of the supertype
- Look for unique attributes and get to the user intent of separate subtypes
- If there are separate add/update transactions with unique attributes for entity subtypes, this *may* indicate separate RETs

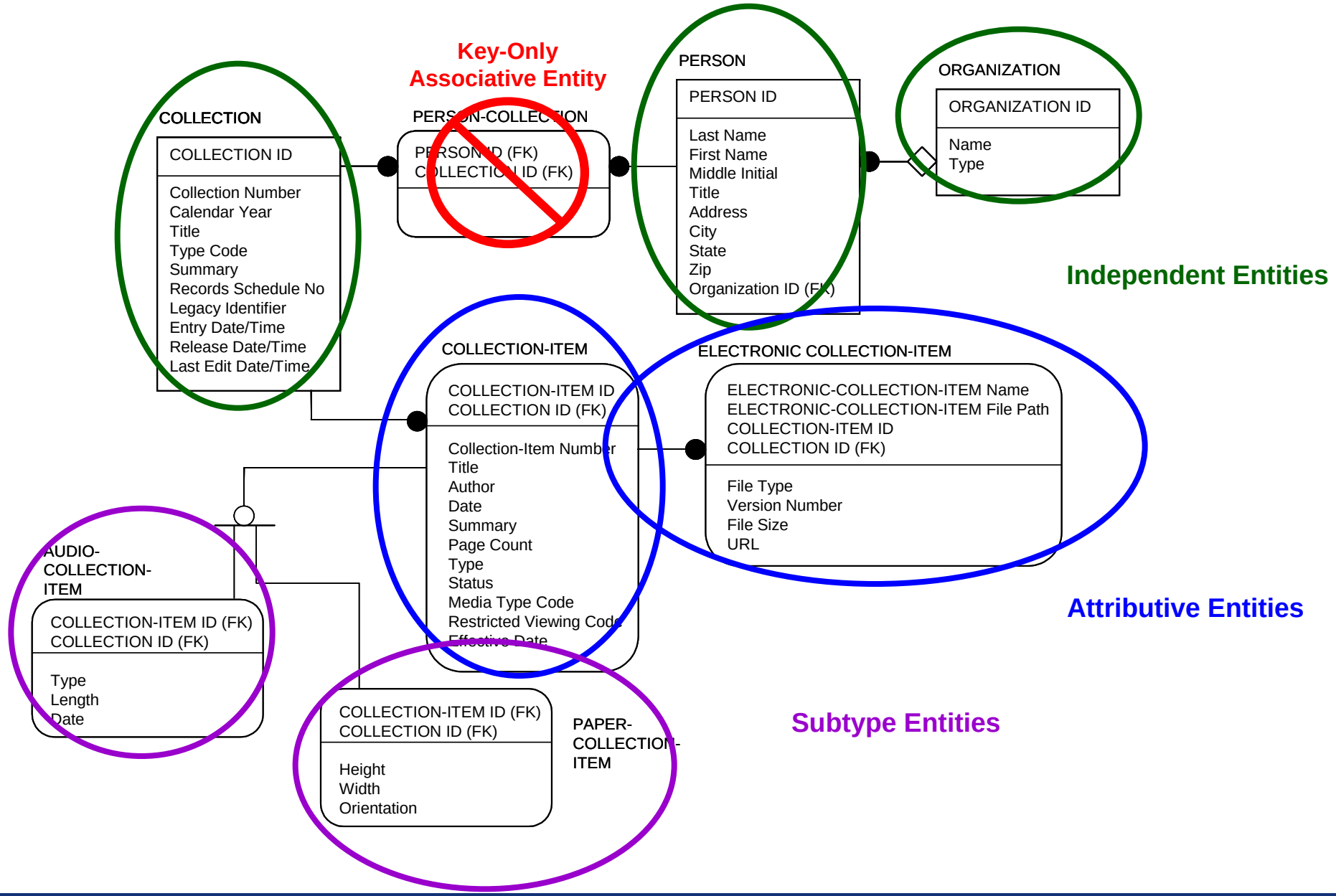


These subtype entities are logical subgroupings that are relevant to the end user, so count as 2 additional RETs

Attributes can give a good indication as to what data element types should be counted in the FPA

③ Identify data element types (DETs)

- *CPM 4.2* rules: count each user recognizable field maintained in or retrieved from a logical file through the execution of an elementary process
- Attributes that are used together in entirety should be counted as a single DET
  - Person name, address are examples of attributes to consider
  - But if in some situations, only parts of the attributes are used, separate DETs might be appropriate
- Look for “sort” or “edit” requirements that would suggest independence from the users perspective
- Make sure attributes are recognized by the user and are not the simply the result of some technical requirement
- Don’t forget foreign keys
  - Count a DET for each piece of data required by the user to establish a relationship with another logical file



Function Point Count, assuming all logical files are maintained by the application

File Type	Name	RETs	DETs	Complexity	UFP
ILF	Organization	1 (*organization)*	*Less than 20*	Low	7
ILF	Person	1 (*person)*	Less than 20	Low	7
ILF	Collection	5 (*collection, collection-item, electronic-collection-item, audio-collection-item, paper-collection-item)*	20-50	Average	10
Total					**24**

Backup Slides

Summary of relationships and counting guidelines

Type of relationship between entities A and B	Conditions	FP Counting Guidelines
(1):(N)	A and B are independent	2 LFs, DETS to each
1:N	B is dependent on A	1 LF, 2 RETS, sum DETS: count each unique, non-repeated field
	B is independent of A	2 LF, DETS: count each unique, non-repeated field for each LF
1:(N)	B is dependent on A	1 LF, 2 RETS, sum DETS: count each unique, non-repeated field1
	B is independent of A	2 LF, DETS: count each unique, non-repeated field for each LF
(1):N	B is dependent on A	1 LF, 2 RETS, sum DETS: count each unique, non-repeated field
	B is independent of A	2 LF, DETS: count each unique, non-repeated field for each LF
(1):(1)	A and B are independent	2 LF, DETS: count each unique, non-repeated field for each LF
1:1	A and B are dependent	1 LF, 2 RETS, sum DETS: count each unique, non-repeated field
1:(1)	B is dependent on A	1 LF, 2 RETS, sum DETS: count each unique, non-repeated field
	B is independent of A	2 LF, DETS: count each unique, non-repeated field for each LF
(N):(M)	A and B are independent	2 LF, DETS: count each unique, non-repeated field for each LF
N:M	B is dependent on A	1 LF, 2 RETS, sum DETS: count each unique, non-repeated field
	B is independent of A	2 LF, DETS: count each unique, non-repeated field for each LF
N:(M)	B is dependent on A	1 LF, 2 RETS, sum DETS: count each unique, non-repeated field
	B is independent of A	2 LF, DETS: count each unique, non-repeated field for each LF

Sources

- Kroenke, David M. *Database Processing: Fundamentals, Design, Implementation, 7th ed.,* Prentice Hall, 2000.
- International Function Point Users Group, *Practical Guidelines for Counting Logical Files, Version 1.0*, September 2001.
- Hay, David C., “A Comparison of Data Modeling Techniques,” Essential Strategies, Inc., 1999.