Logical E/R Modeling: the Definition of ‘Truth’ for Data

Jeff Jacobs
Jeffrey Jacobs & Associates
Belmont, CA
phone: 650.571.7092
email: jeff@jeffreyjacobs.com
http://www.jeffreyjacobs.com

Survey

- How important is data to your organization?
- Do you have an organization responsible for enterprise data?
- Do you use RDBMS?
- Are you using UML?
- Does your organization have a methodology or process, such as RUP?

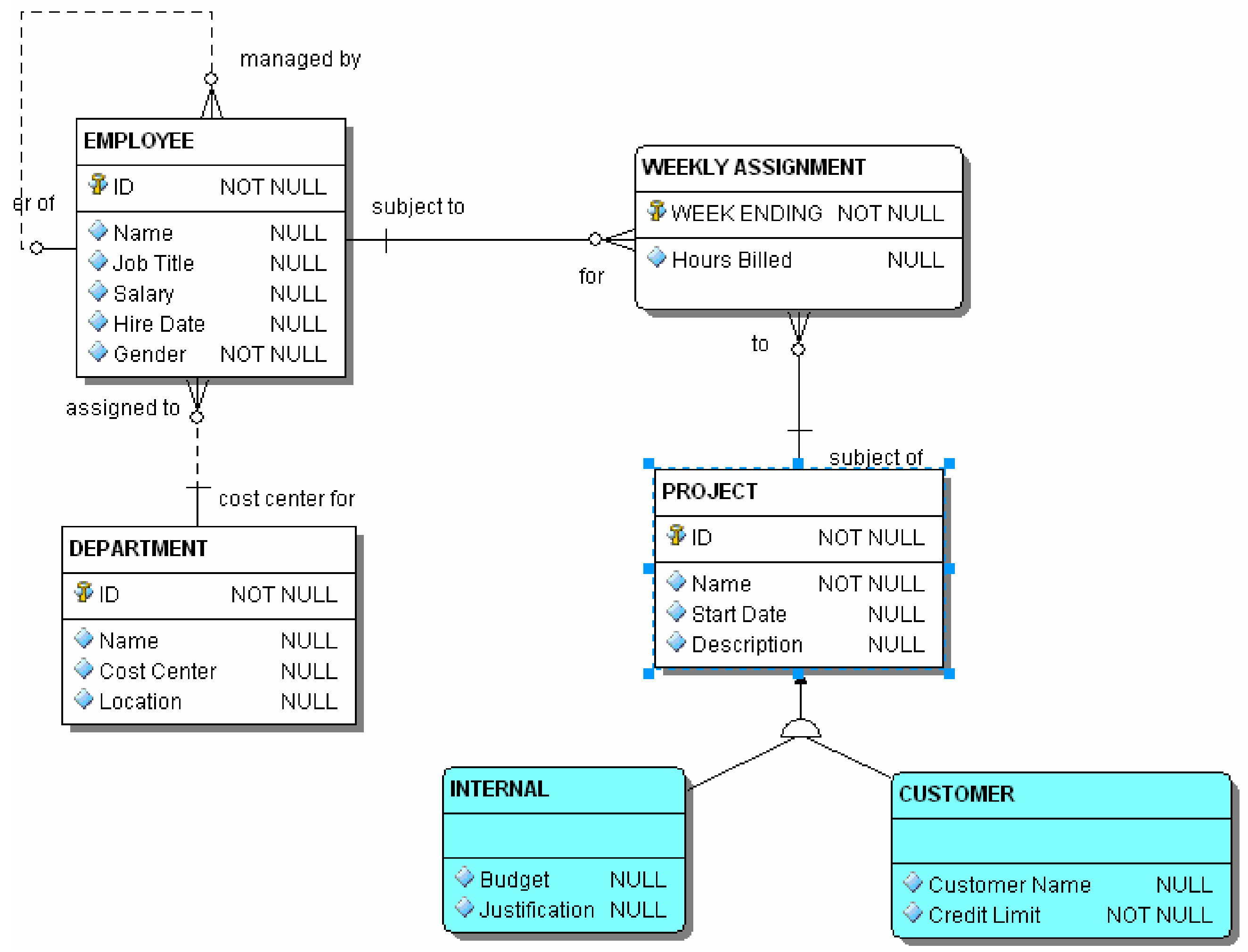
Objective

- Learn the fundamentals of Entity Relationship modeling
- Why?
  - Improve overall quality of product requirements
  - Ensure that all necessary data is present for all areas of products, including reporting
  - Understand the business requirements
  - Provide basis for implementation
  - Provide basis for UML class model

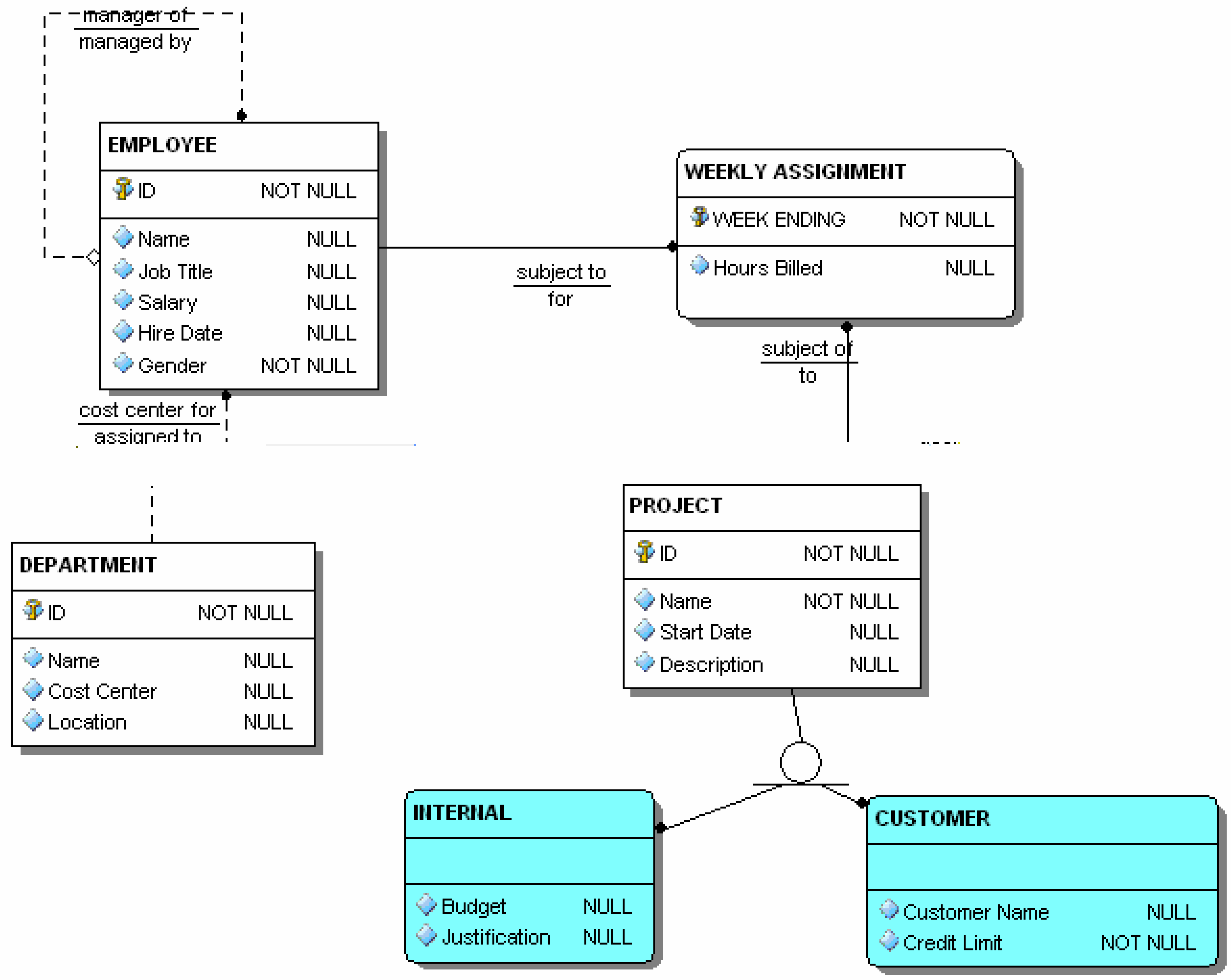
Introduction to Entity Relationship Modeling

- ER modeling establishes the “information requirements” of the business, e.g. *What information must be kept to meet the functional requirements*
- An ER model consists of definitions of *entities*, *attributes*, *relationships*, *domains* and supporting detailed information
- An ER Diagram (ERD) is a “picture” of the model
- Numerous notations include Information Engineering (IE), IDEF1X, Oracle, Chen, UML (?)
- Many tools have their own variation

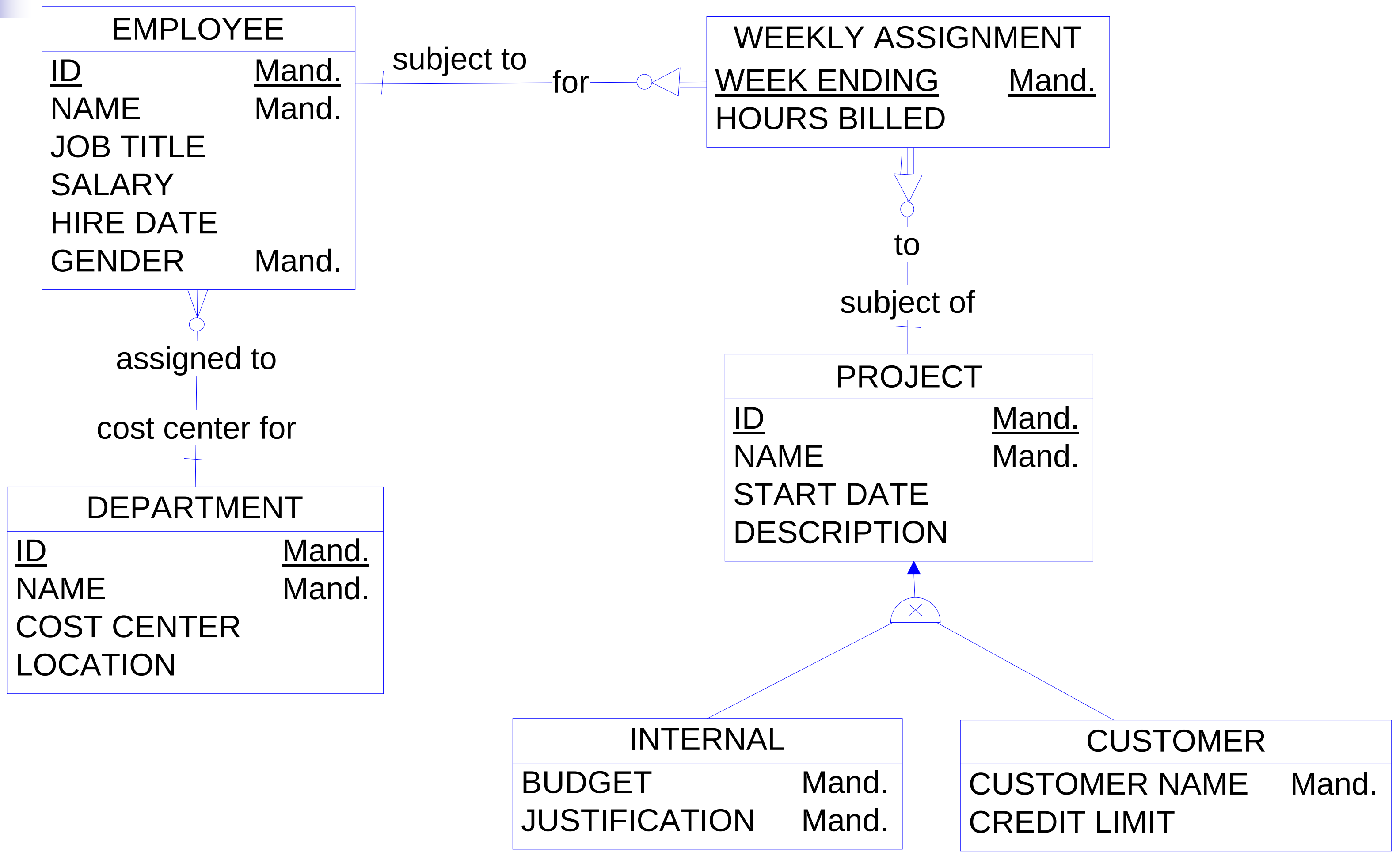
ERD Example (Information Engineering)



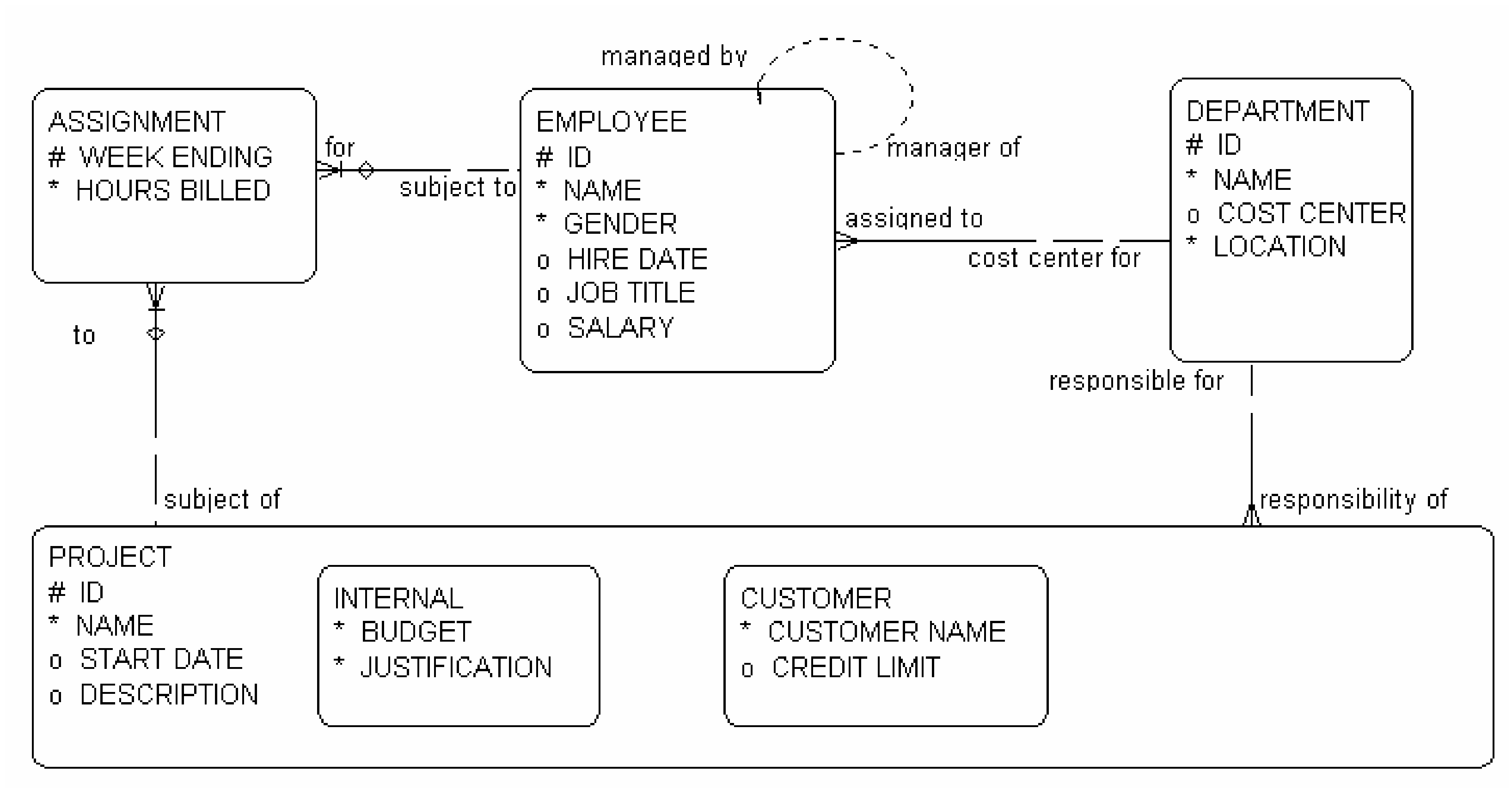
IDEFI1X Notation



ERD Example (PowerDesigner "IE")



ERD Example (Oracle)



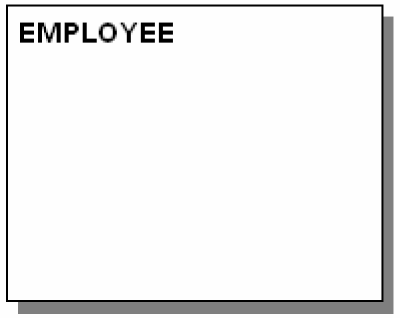
ER Modeling is “Semantic”

- ER modeling establishes information and data requirements, without regard to the eventual implementation
  - implementation may be relational database, object stores, in-memory data or even paper
  - typical implementation is relational database
- Also called “Semantic Data Model”
- Sometime called “Conceptual”
- “Physical Data Model” (PDM) has additional information for generating relational database; diagrams are similar
- Disagreement over “Logical Data Model”…

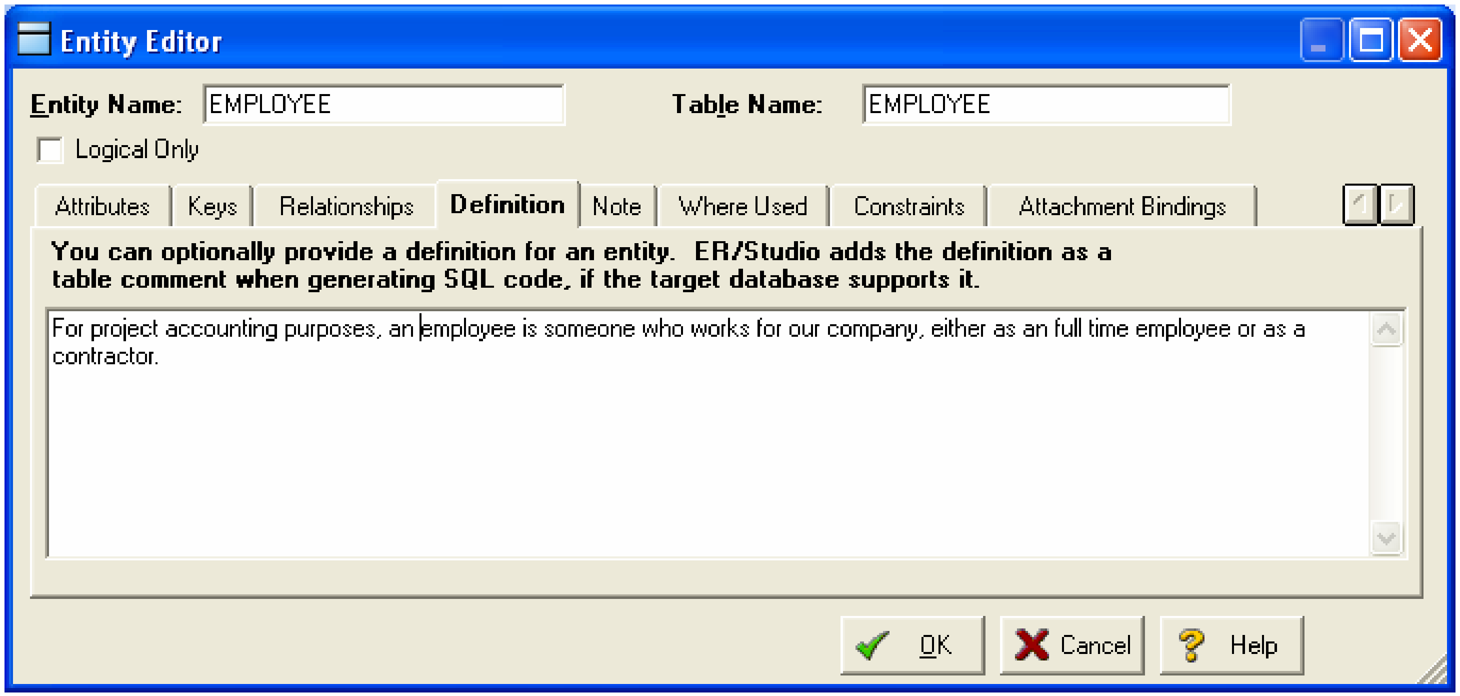
Entities

- “A thing of significance in the business about which information must be kept and maintained”
- Entity name is always singular
- Entity name is meaningful to the business, part of common vocabulary
- 2 main categories of information
  - 1) Attributes
  - 2) Relationships with other entities
- Drawn as square box
- Additional information depends on tool, methodology

Entity Example



Supporting Information



Entity Name: EMPLOYEE
Table Name: EMPLOYEE

☐ Logical Only

You can optionally provide a definition for an entity. ER/Studio adds the definition as a table comment when generating SQL code, if the target database supports it.

For project accounting purposes, an employee is someone who works for our company, either as an full time employee or as a contractor.

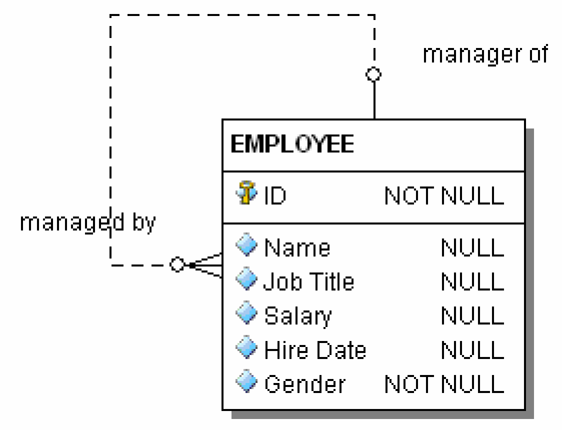
Instances and Occurrences

- Entity definition is a “type” or “class” description, e.g. EMPLOYEE
- Don’t confuse entity “type” with “occurrence/instance” of an entity, e.g. Joan Smith is an occurrence of the entity type EMPLOYEE

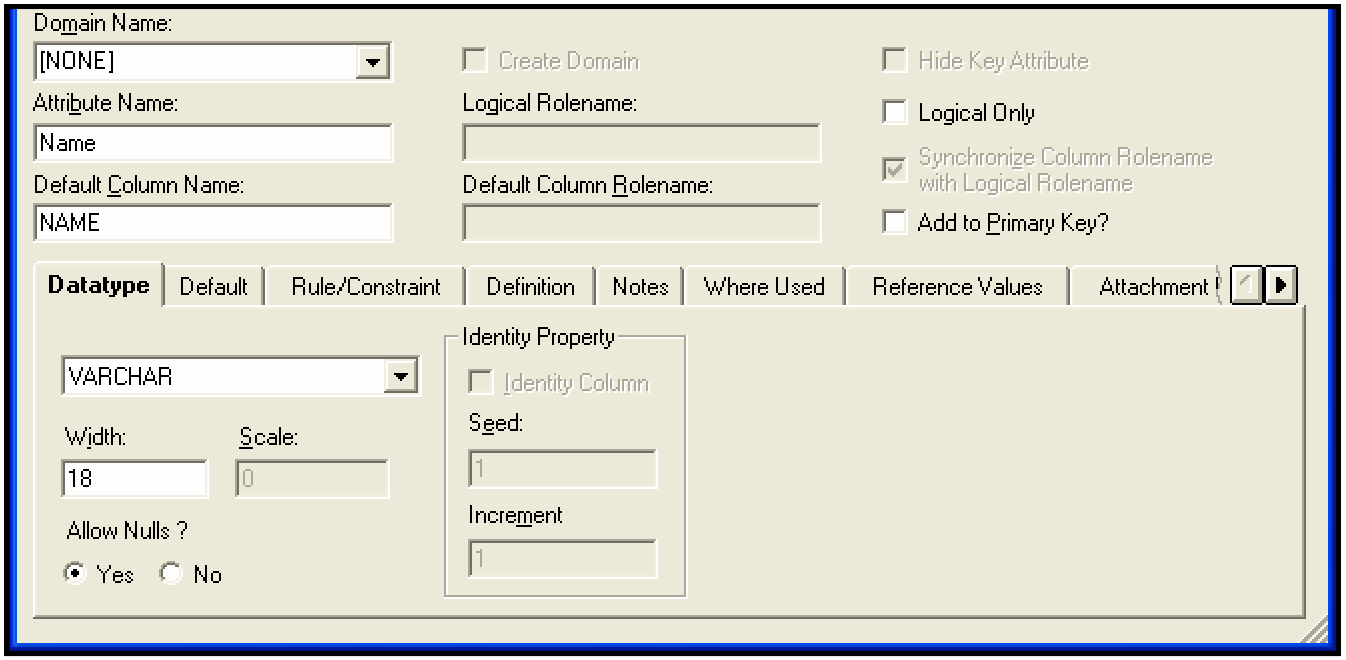
Attributes

- "Individual, atomic pieces of data *about* an entity"
- Never used to refer to another entity. (Attributes are *not* foreign keys in ER)
    - Some notations/tools show "foreign keys"
- Attributes may be mandatory or optional
- Mandatory means "*every instance of the entity must, at all times, with no exceptions of any duration, have a valid, non-NULL value for the attribute*"
- Optional means "*value may sometimes by undefined or unknown*", NULL
- Usually indicated by "Not Null" or "Mand" or symbol; depends on tool

Entity Example with Attributes



Attribute Supporting Information



Domain Name: [NONE]

☐ Create Domain

☐ Hide Key Attribute

Attribute Name: Name

Logical Rolename:

☐ Logical Only

☑ Synchronize Column Rolename with Logical Rolename

Default Column Name: NAME

Default Column Rolename:

☐ Add to Primary Key?

VARCHAR

Width: 18

Scale: 0

Allow Nulls ?

[✓ Yes] No

Identity Property

☐ Identity Column

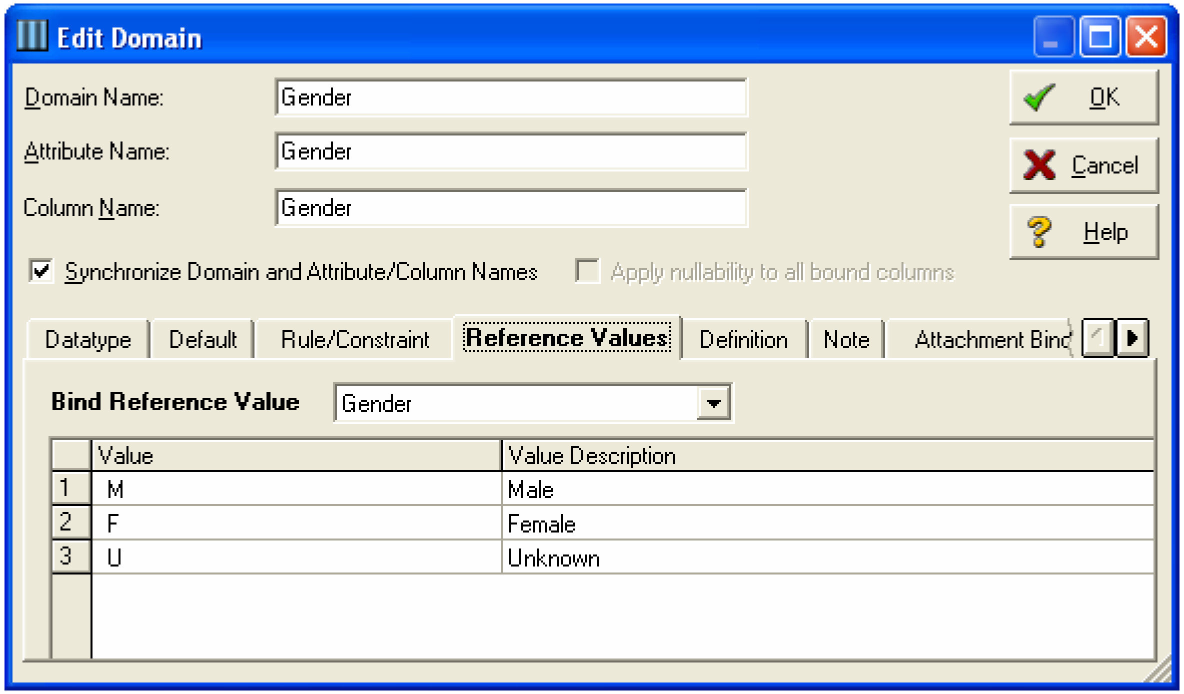
Seed: 1

Increment 1

Domains

- Domain is a centralized definition of valid values and datatype information for attributes (and columns)
- Attributes that “belong” to a domain inherit the characteristics of the domain, e.g. datatype information and allowable values
- Example: “Gender” domain has datatype of VARCHAR2(1) and valid values of [M|F|U] with meanings of “Male”, “Female”, “Unknown”
- Attributes can have the same name as the domain to which they belong
- Domains may be “nested”, providing levels of validation, e.g. the SALARY domain belongs to the MONEY domain
- No ”diagramming” technique for domains
- Domains usually result in column constraints or reference tables/classes

Domain Definition



Domain Name: Gender

Attribute Name: Gender

Column Name: Gender

[✓] Synchronize Domain and Attribute/Column Names [ ] Apply nullability to all bound columns

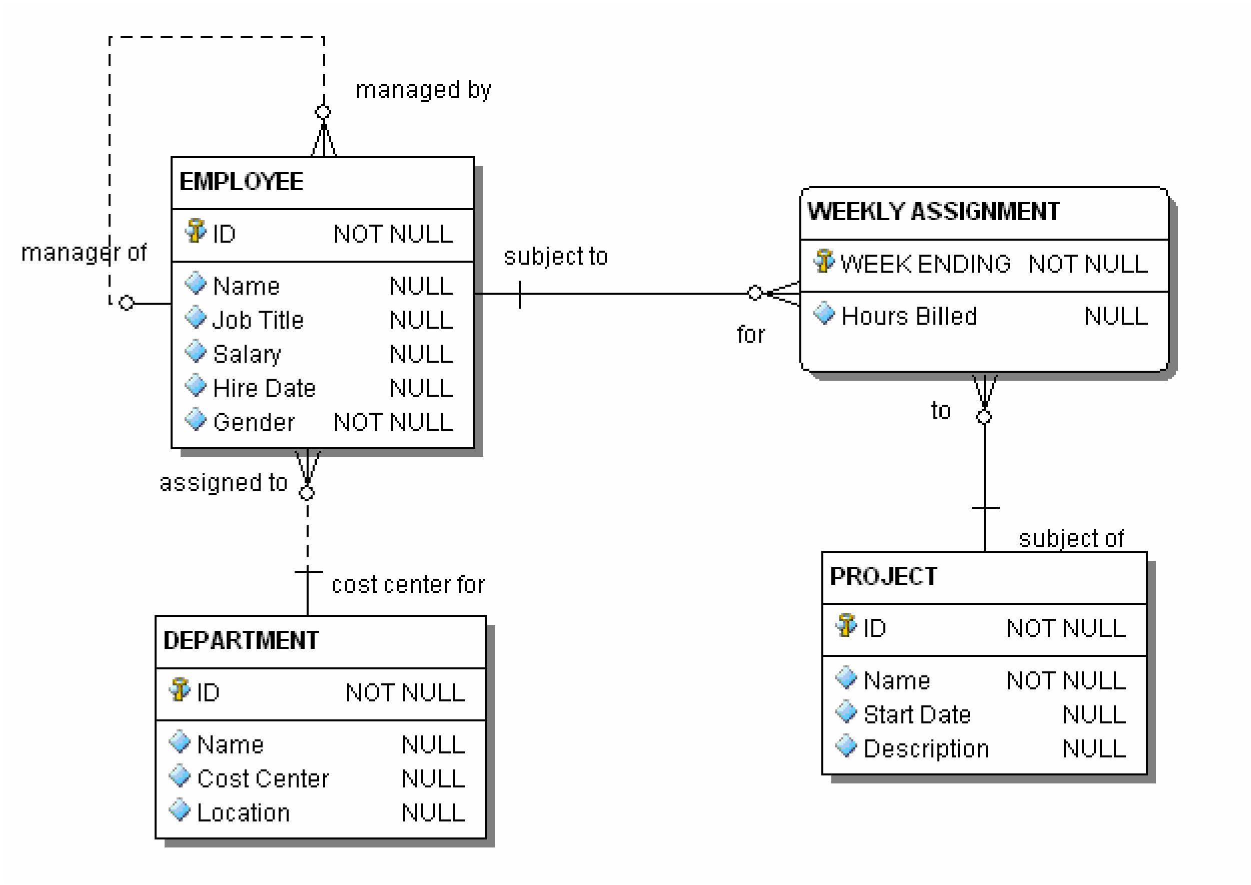
Bind Reference Value Gender

	Value	Value Description
1	M	Male
2	F	Female
3	U	Unknown

Relationships (not Relations)

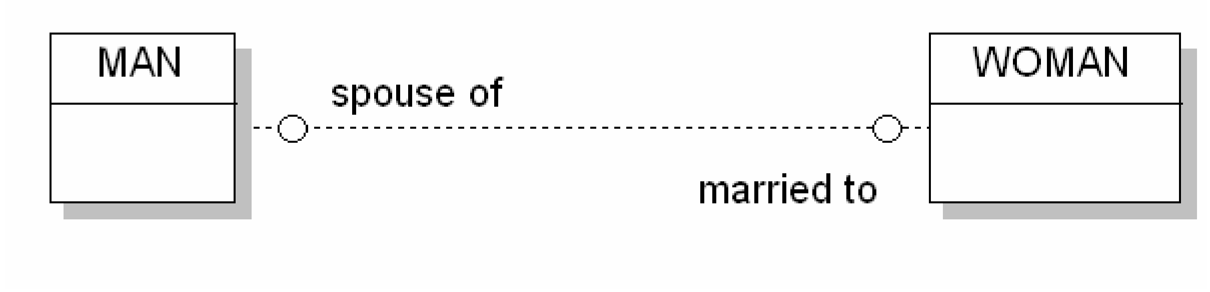
- "A named/labeled association between two entities" (drawn as a line)
- Two names for a relationship, one for each direction
- Naming is very important
  - critical to understanding
  - defines *semantics* in resulting implementation in business terms
- Tools typically support additional definition, notes, etc.

Entities with Relationships Example



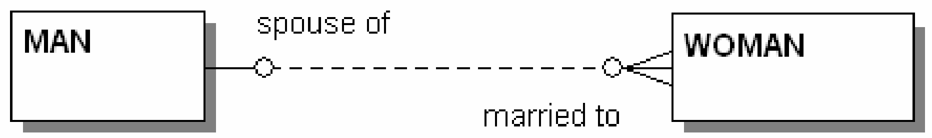
Optionality

- Relationships have *optionality* expressed as either *mandatory* or *optional* in each direction
- "Mandatory" means "*every occurrence of the entity must always, at all times, <u>with no exceptions of any duration</u>, be associated with an instance of the entity at the other end*"
- "Optional" means "need not always be associated…"



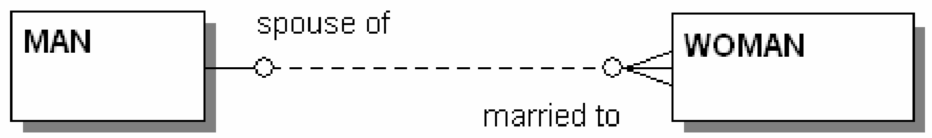
Cardinality

- (Maximum) Cardinality comes in two flavors
  - 1) "*One and only one*", e.g. each occurrence of an entity may be associated with at most one occurrence at the other end; optionality determines if such an association must exist
  - 2) "*One or more*", e.g. each occurrence may be associated with zero (depending on optionality), one or more occurrences at the other end



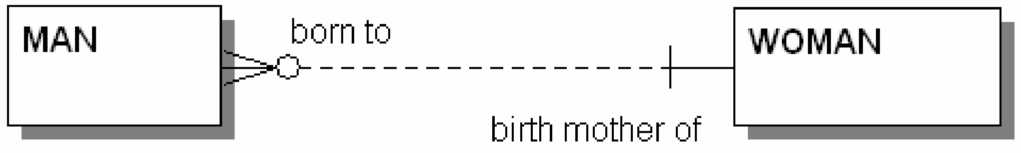
Reading Relationships

- Proper reading of relationship contributes to reliability and confidence
- Relationships should be understandable in both directions
- Reading starts with <entity1>
- EACH <entity1> [MAY BE | MUST BE] <rel1> [ONE OR MORE | ONE AND ONLY ONE] <entity2>
- "EACH" reminds us that we are talking about instance/occurrences of entities



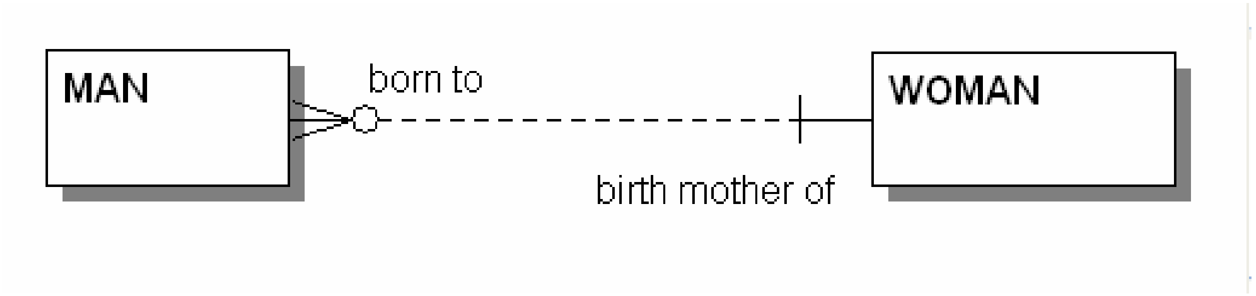
Optionality

- [MAY BE | MUST BE] expresses optionality; more understandable than “zero”
  - ‘o’ on line at end opposite <entity1> is “MAY BE”
  - ‘|’ on line at end opposite is “MUST BE”
- [ONE OR MORE | ONE AND ONLY ONE] expresses maximum cardinality
  - *presence* of crow’s foot at end opposite <entity1> is “ONE OR MORE”
  - *absence* of crow’s foot at end opposite <entity1> is “ONE AND ONLY ONE”



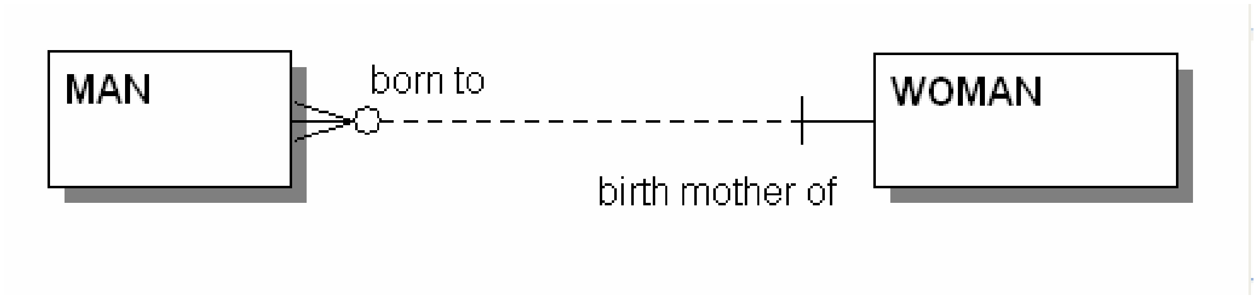
Relationship

- Read relationship name *adjacent to entity1*
- Example: EACH ENTITY1 MUST BE <rel1> …



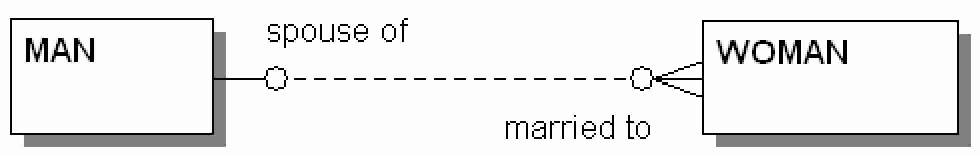
Cardinality

- Look at symbols on line adjacent to <entity2> to determine cardinality
- [ONE OR MORE | ONE AND ONLY ONE] expresses maximum cardinality
  - 1) presence of crow’s foot at end adjacent to <entity2> is “ONE OR MORE”

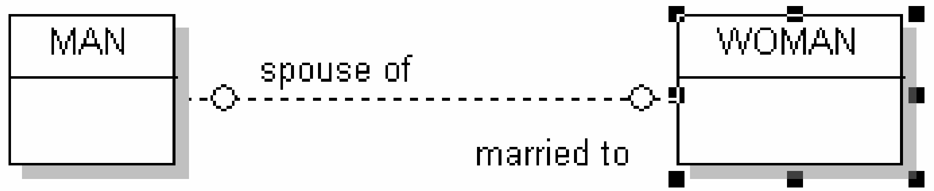


Cardinality

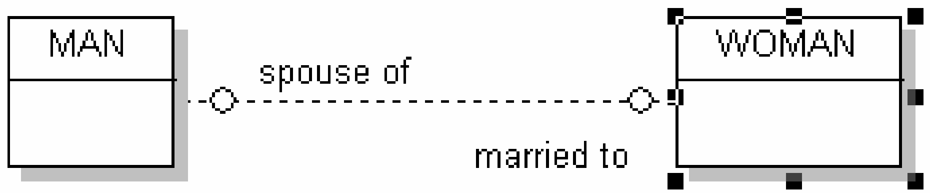
- Look at symbols on line adjacent to <entity2> to determine cardinality
- [ONE OR MORE | ONE AND ONLY ONE] expresses maximum cardinality
  - 1) presence of crow's foot at end adjacent to <entity2> is "ONE OR MORE"

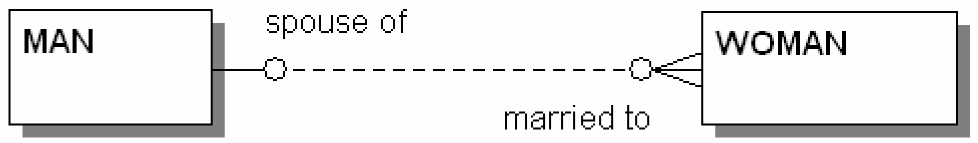


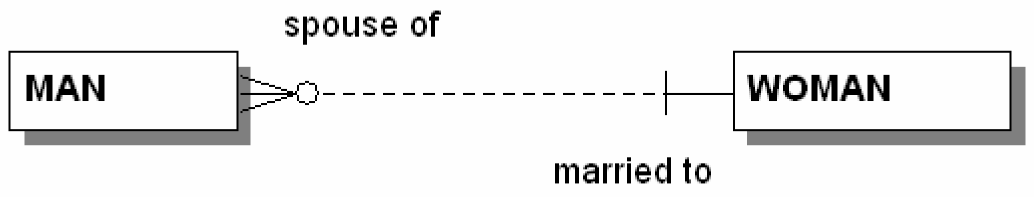
  - 2) absence of crow's foot at end adjacent to <entity2> is "ONE AND ONLY ONE"



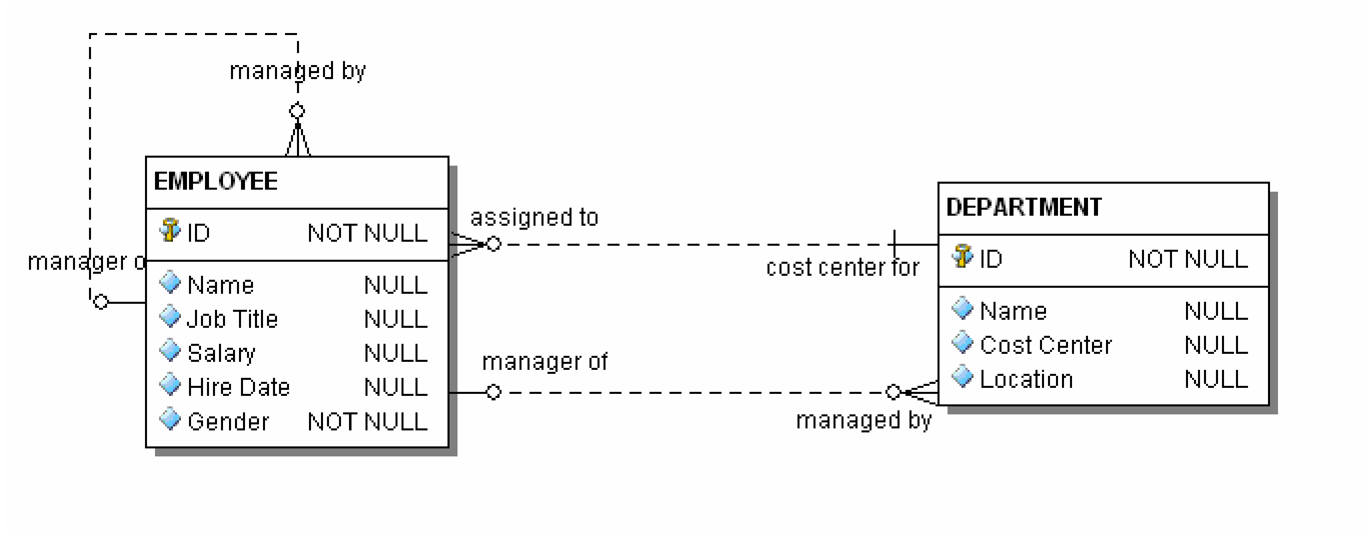
Anthropological Examples





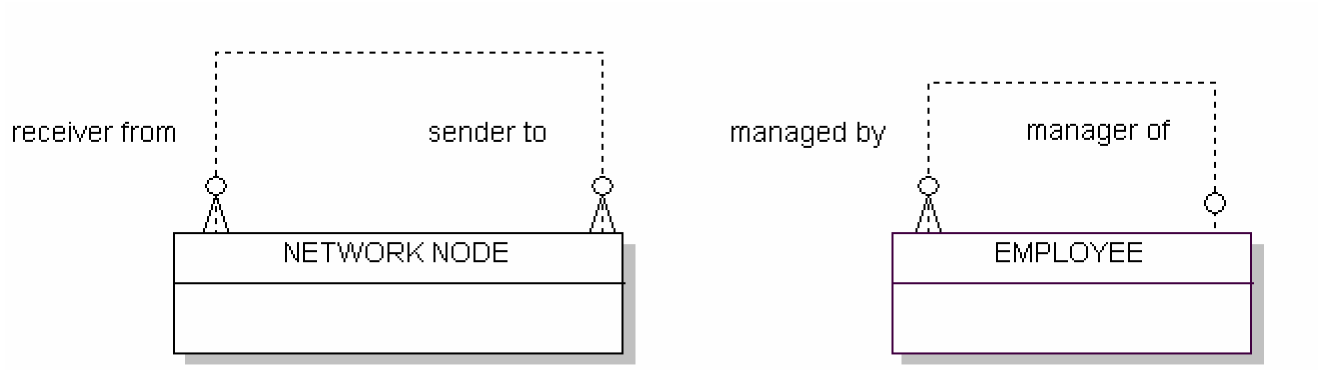


Multiple Relationships Between Entities

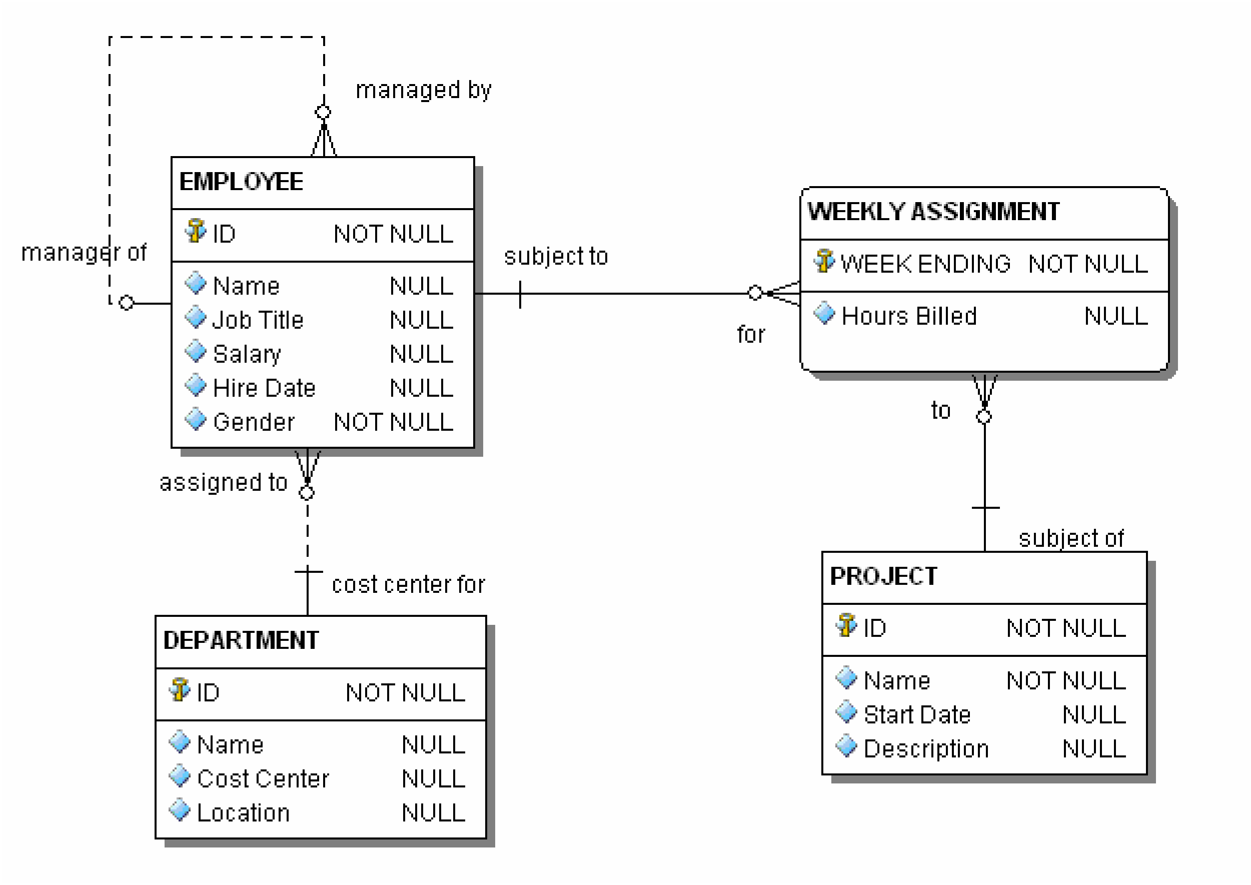


Reflexive/Recursive Relationships

- Relationships may exist between occurrences of the same entity type
- Recursive relationships are used for hierarchies and networks
- Must be optional in both directions



Relationship Reading Exercises



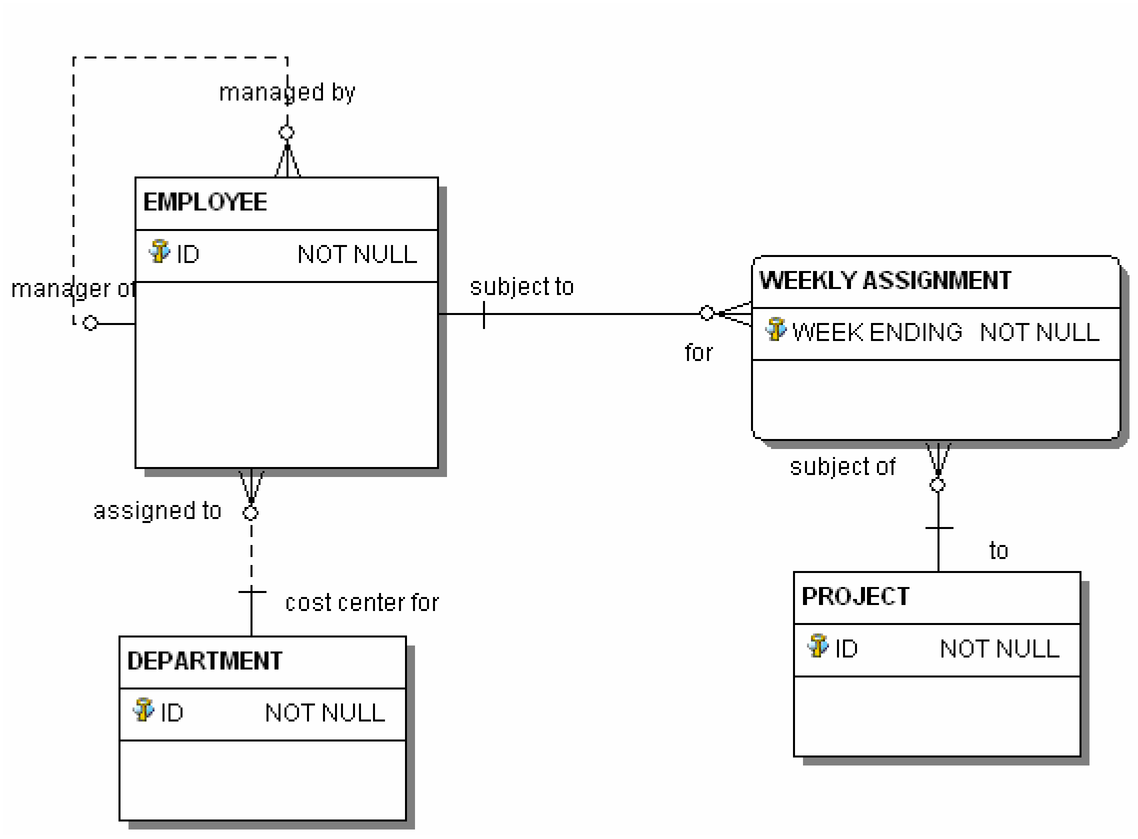
Unique Identifiers (UID)

- A combination of attributes and/or relationships used to uniquely identify and distinguish each occurrence of an entity from all others
- No two occurrences may have the same set of values for all parts of the UID
- UID may consist of
  - 1) single attribute
  - 2) multiple attributes
  - 3) multiple relationships
  - 4) combination of attribute(s) and relationship(s)

More UID

- All parts of UID must be mandatory
- Attributes that are part of UID are typically in their own “box” with tool specific indicator
- Relationships in a UID are indicated by a solid line in IE and IDEF1X; different in some tools (PowerDesigner and Oracle)

UID Examples

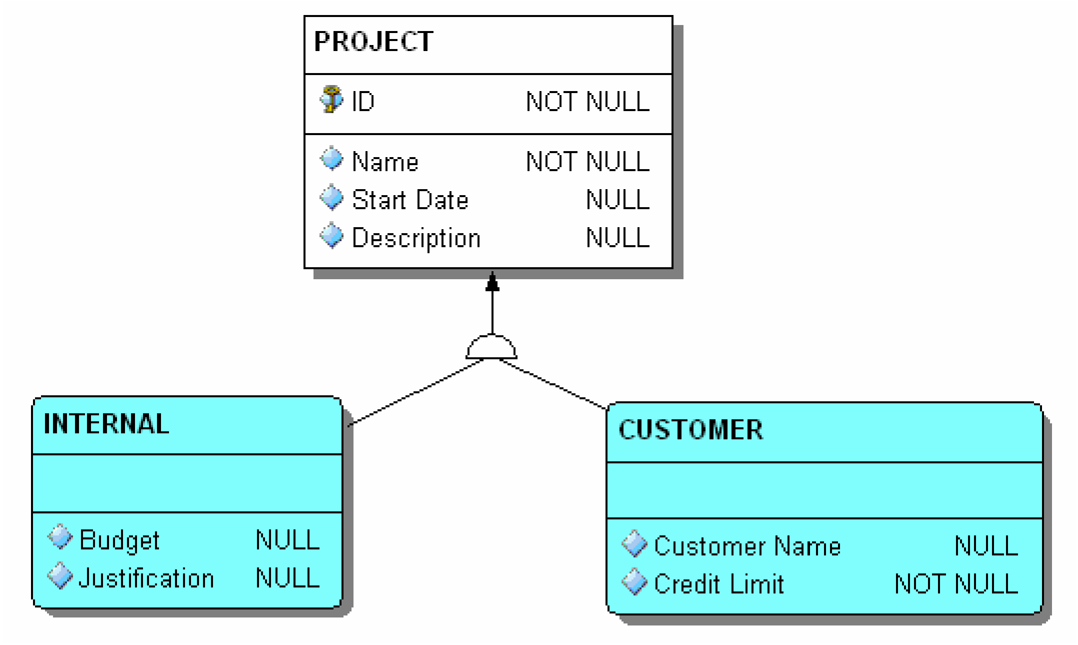


Super/subtype Entity Structure

- Subtype entities are “specializations” of supertype entity
- Subtype inherits all attributes and relationships of supertype entity
- Occurrence of subtype is also occurrence of supertype; UID is always at the outermost supertype
- Each subtype should (eventually) have its own attributes and/or relationships
- Subtypes are “exhaustive”; all occurrences of the supertype must also be occurrence of a subtype
- Subtypes are “exclusive and non-overlapping”; no occurrence can belong to more than one subtype (except via “nesting”)
- Subtypes may be “complete” or “incomplete”
  - Complete means all subtypes are known

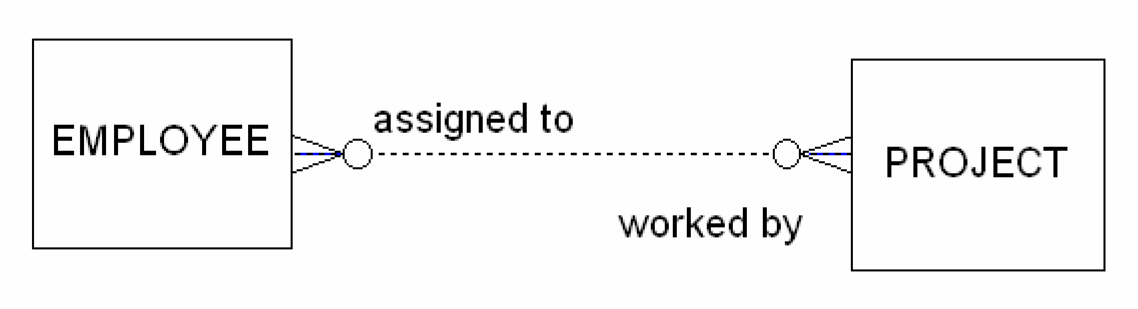
Subtype

- Use special connectors
- May allow specification of exclusive/non-exclusive (aka overlapping/non-overlapping)
- May allow specification of complete/incomplete



Many to Many (M:M) Relationships

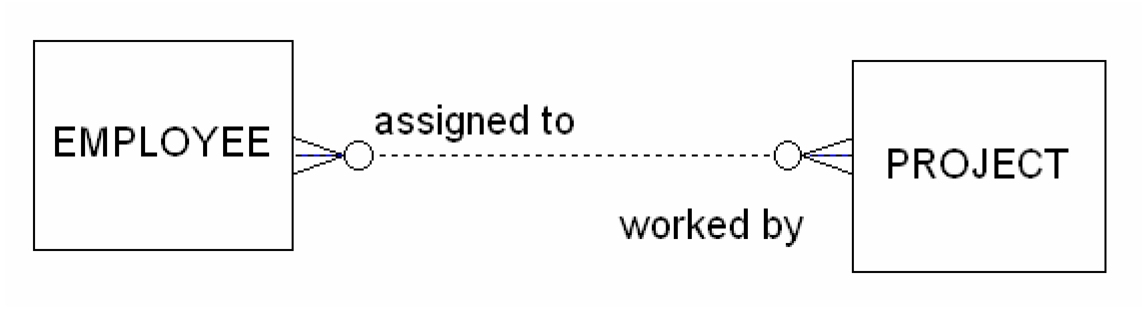
- The start of “conceptual modeling”
- M:M relationships “hide” important detail that must be discovered
- M:M relationships should be eliminated by end of detailed requirements analysis
- Iterative process of refinement



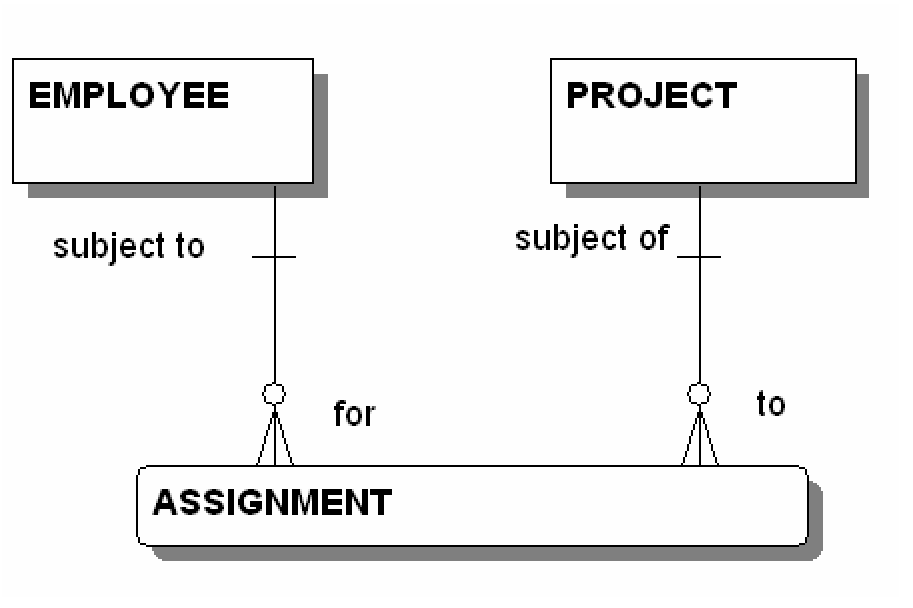
Resolving

- To resolve a M:M relationship:
  - 1) Create new entity
  - 2) Create relationships back to original entities
  - 3) Include relationships as part of new entity’s UID
  - 4) Use meaningful names for new entity and relationships
  - 5) Examine new entity for attributes and relationships
- *Major “flaw” in UML modeling techniques*

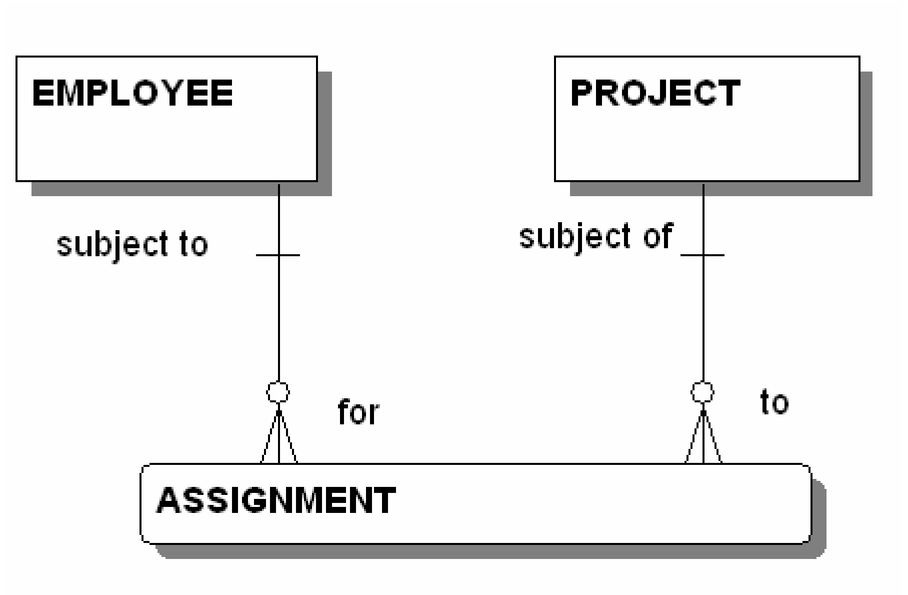
Resolving M:M



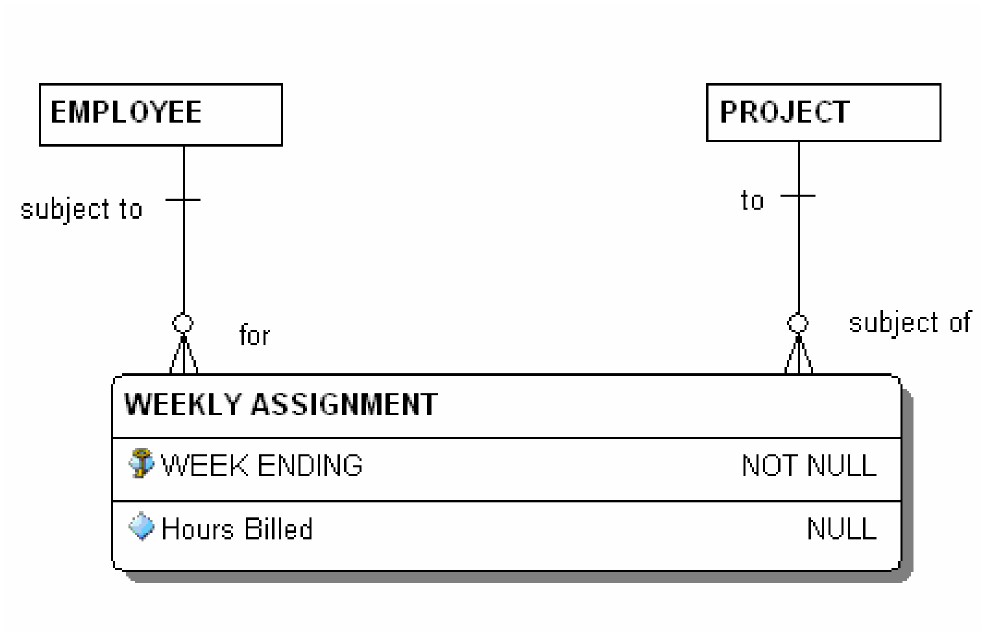
- Create new entity with UID/dependant relationships
- New name is important! Not “EMP/PROJ”!!!



- Re-examine for new attributes and relationships



- Re-examine for new attributes and relationships



QA’ing ERDs

- For each entity:
  - Is the name precise?
  - Is the name a recognized business term?
  - Is the name singular?
  - Can the name be improved?
- For each attribute:
  - Is the attribute name precise?
  - Is the attribute name understandable?
  - Is the name a a recognized business term?
  - Is the optionality correct?
  - Can the name be improved?

QA of Relationships

- For each relationship:
  - Is the optionality correct? (If it's mandatory, are there any exceptions?)
  - Is the cardinality correct?
  - Is the name precise and meaningful? (*Very important!!!)*
  - Is the name a recognizable business term?
- *Can you find all of the information you need for your development area?*

Summary

- ERD captures information requirements
- Proper reading eliminates ambiguity
- ERD should be understood by all interested parties
- Wording and terminology is critical