Data Structure Patterns

Additional Material (not in course textbooks)

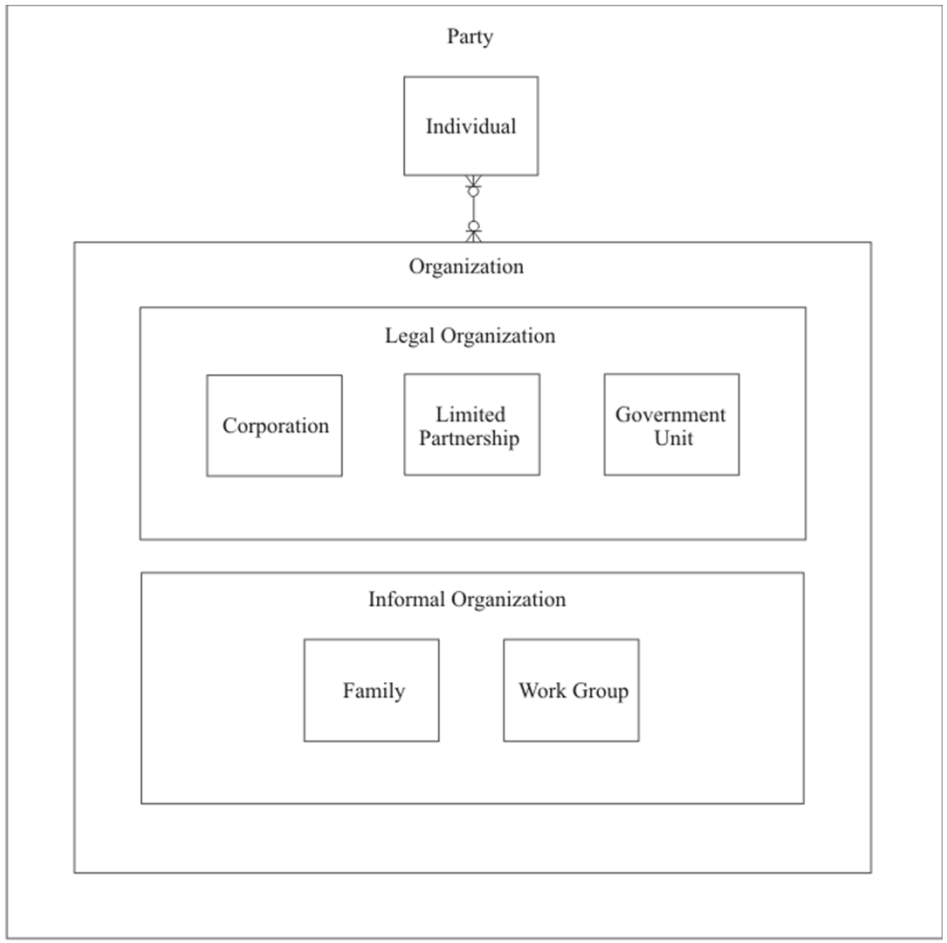
Data Structure Patterns

- Very few data modeling projects require you to start with the proverbial blank slate.
  - Look for similar existing conceptual models.
  - Look for common patterns that have previously worked.
  - Look for useful generic models and patterns.
- However, make sure you ***adapt*** the pattern model rather than force fitting it.

The Party Model

- A *party* is a person or organization of interest to the database application.
- The party model adds Party as a supertype.
  - This provides the ability to hide the party's implementation details in cases where it doesn't matter.
  - For example, when modeling tasks, we need only a single relationship to assign tasks to parties without regard for whether the party was an individual or organization

Party Model Data Pattern

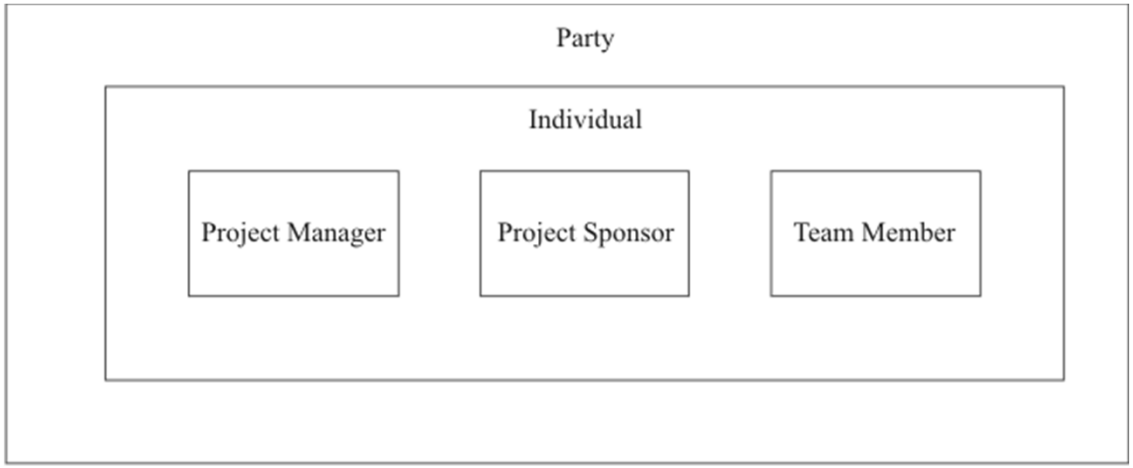


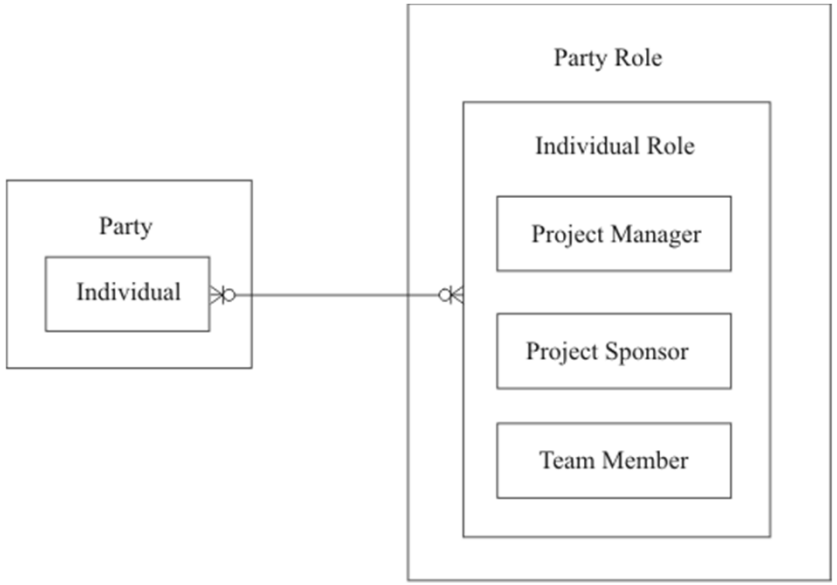
- Subtypes need to be tailored the specific organization
- The subtype Individual has a M:N relationship with the subtype Organization.
  - This unusual construct that may be better as a role (covered next).

The Role Model

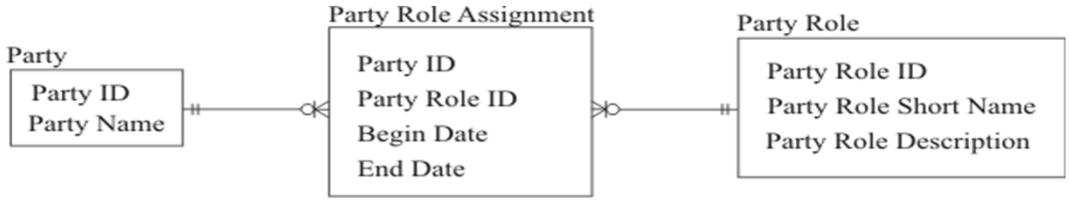
- An alternative to supertypes and subtypes
- Works better than subtypes when:
  - Subtypes overlap (one supertype occurrence can be many subtypes)
  - Subtypes are frequently added and removed
    - A role addition requires only an INSERT to a table (rather than a table CREATE)
    - A role removal requires only a DELETE to a table, or an UPDATE of the effective dates of an existing row for the role.

Subtype vs. Role Example





Party Role Logical and Physical Examples



Party

Party ID	Party Name
101	Tiana C. Steger
102	Andrew T. Guay
103	Sherry L. Johnson

Party Role Assignment

Party ID	Party Role ID	Begin Date	End Date
101	PS	01/05/2009	12/31/2010
101	PM	01/05/2009	06/30/2009
102	TM	01/05/2009	12/31/2010
103	PM	07/01/2009	12/31/2010
103	TM	07/01/2009	12/31/2010

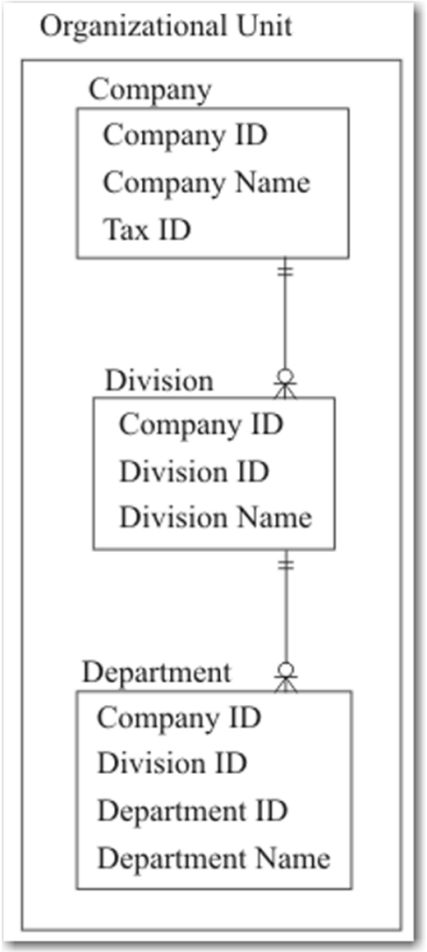
Party Role

Party Role ID	Party Role Short Name	Party Role Description
PS	Prj Spnsr	Project Sponsor
PM	Prj Mgr	Project Manager
TM	Mbr	Project Team Member

The Generalized Hierarchy Model

- *Hierarchies* are structures where entities are organized into layers in which each entity:
  - Can have any number of children (subordinate entities) at lower layers
  - Can have only one parent (superior entity) at the next higher layer.
- Sometimes called tree structures because they look like an upside-down tree
  - Topmost entity is called the root
  - Bottommost entities called leaves

Organizational Hierarchy Example



Company

Company ID	Company Name	Tax ID
M100	Acme Industries	07-7542678
M110	Northwest Manufacturing	07-9426104

Division

Company ID	Division ID	Division Name
M100	01	Rocketry
M100	02	Safety Equipment

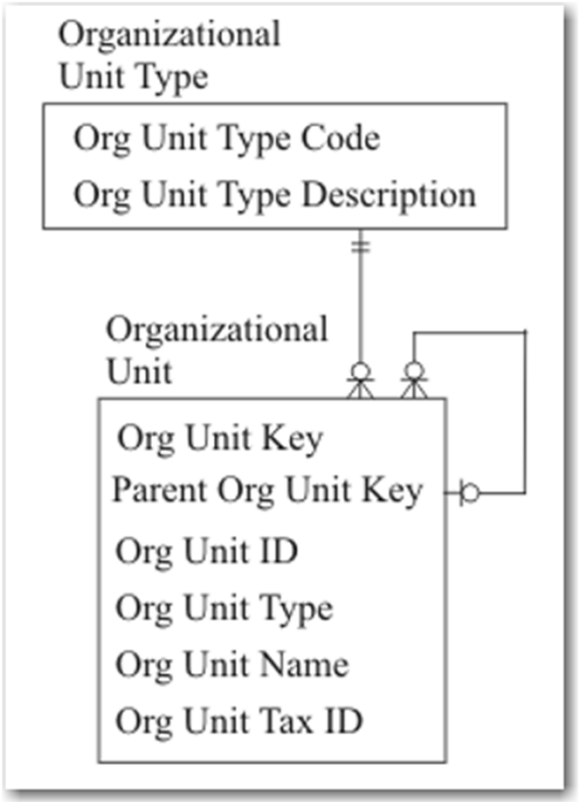
Department

Company ID	Division ID	Department ID	Department Name
M100	01	M1	Manufacturing
M100	01	S1	Government Sales
M100	01	S2	Non-Government Sales
M100	02	S1	Government Sales
M100	02	S5	Retail Sales

Specialized Hierarchy Weaknesses

- To add another layer, a new table must be created
- If some companies in the database have departments but no divisions, we will have to add a dummy division in their hierarchy to link their departments to the company layer.
- The structure can be confusing to use if parts of the company use different names for the layers.

Generalized Hierarchy



Organizational Unit Type

Org Unit Type Code	Org Unit Type Description
CO	Company
DIV	Division
DEPT	Department

Organizational Unit Type

Org Unit Key	Parent Org Unit Key	Org Unit ID	Org Unit Type Code	Org Unit Name	Org Unit Tax ID
1001		M100	CO	Acme Industries	07-7542678
1002	1001	01	DIV	Rocketry	
1003	1002	M1	DEPT	Manufacturing	
1004	1002	S1	DEPT	Government Sales	
1005	1002	S2	DEPT	Non-Government Sales	
1006	1001	02	DIV	Safety Equipment	
1007	1006	S1	DEPT	Government Sales	
1008	1006	S5	DEPT	Retail Sales	
1009		M110	CO	Northwest Manufacturing	07-942104

Generalized Hierarchy Usability

- Advantages:
  - Can handle any number of layers and any type of organizational unit.
  - Layers do not have to be uniform
- Disadvantages
  - More difficult for concrete thinkers
  - Must use generic identifiers
  - Some designers will generalize for the sake of generalizing
    - Sometimes technically sophisticated solutions are not the smartest choice