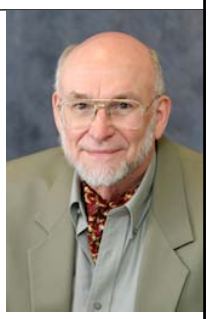
DAMA-Phoenix, 2009/09

35START

1



Subtypes & Supertypes

The Data Modeler's most Valuable Construct

©Gordon C. Everest
Professor Emeritus of MIS and Database
Carlson School of Management
University of Minnesota
geverest@umn.edu

Data Modeling Constructs

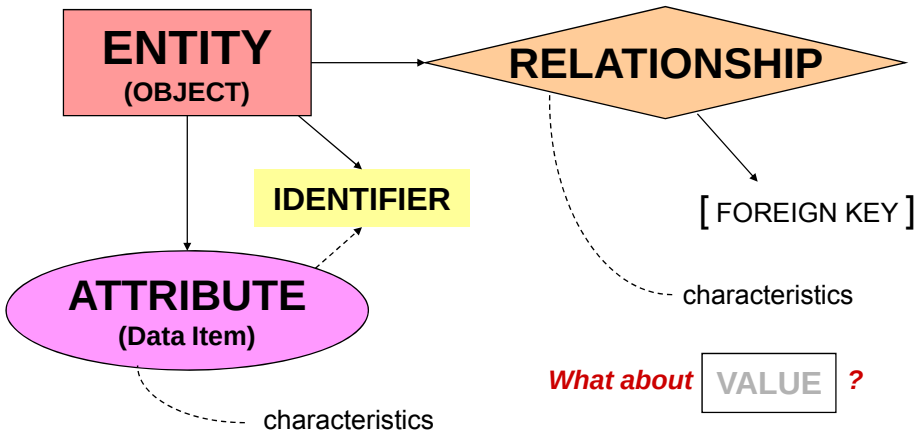
DMOD

2

What to look for:

Relative emphasis differentiates Data Modeling Schemes

e.g. ER modeling focuses on Entities and Relationships, de-emphasizing, even hiding Attributes.



Advanced Database Design

431START

3

7. Sub/Super Types

- "Abstractions" v. Collections v. Generalization
- Attribute, Relationship, & Entity Generalization
- Subtypes and Supertypes (Entity Generalization)
  - Underlying assumptions; conditions
  - Generalization vs. Specialization
  - Diagramming - Graphical representations
- Constraints
- Subtype Definition - distinguishing attribute
- Sub/SuperType Hierarchy/Lattice
  - The Universal Relation
- Inheritance (single; multiple) & Reuse
- Mapping to Tables

©Gordon C. Everest
Carlson School of Management
University of Minnesota

Starting from Basics

G

4

When you see this in an ER/schema diagram:

CUSTOMER **EMPLOYEE**

What does it mean?

What can you assume?

What does "the system" assume?

At least four
distinct (context-free)
semantic statements.

N

A Model

DMOD

5

A Model is an *abstract* *re.presentation*

... of something observed or something to be built.

Some "reality" (in some domain of interest) *presents* itself to the modeler.
The modeler builds a model of what they perceive or design.
Hence, the model is a *re.presentation* of that reality.

Abstract because some aspects of the "reality" are *omitted*
(necessarily, because people are doing the observing or designing
and therefore it is an imperfect process).

The *semantics* of a model are seen through some *syntax*.
The *syntax* is the chosen method/scheme for representation.

Abstraction

DMODPRE

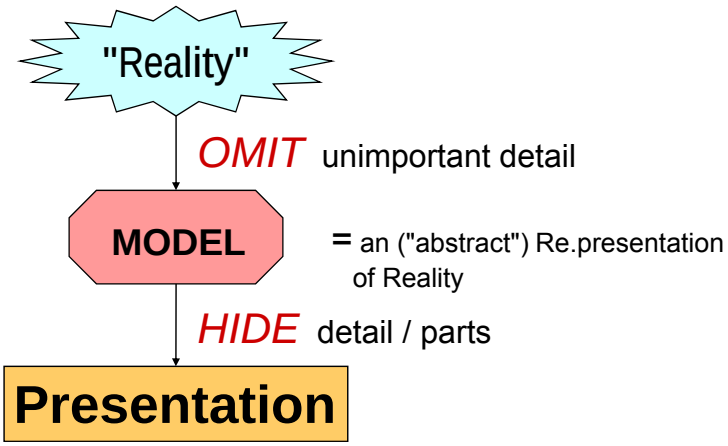
6

Jeff Kramer, *CACM* (50:4) 2007 April, p.36
www.cse.msu.edu/~cse914/Papers/Kramer-abstraction.pdf

- Definition - used in two senses:
  - (1) removing detail
  - (2) generalization - to identify commonalities
- Fit for a Purpose
- Widely used in art, music (jazz), maps (London Underground)
- Essential to Computing, (Software) Engineering, Requirements elicitation, (Data) Modeling
- Fourth level of human development (J. Piaget, 1896-1980)
  - "formal operational"- ability to think abstractly, systematically, hypothetically, symbolically
  - only one third achieve this level; some people never reach it! They do not appreciate the value of modeling; and find it difficult to identify what is important in a problem.
- Need a test to measure abstraction ability

Abstraction: Omitting vs. Hiding

DMODPRE
SSTYPE
7



"There is no abstract art. You must always start with something. Afterward you can remove all traces of reality." -- Pablo Picasso

"Abstractions" & Collections

SSTYPE
8

Focusing on *selected* properties of objects

- Batini, Ceri, & Navathe
- Smith & Smith (1977)
- Graeme Simsion
- Steve Hoberman
- Len Silverston
- David Hay

"Abstractions": (used in a different sense here)

- CLASSIFICATION ("Member-of")
  - Forming Types - entity sets/populations, domains
- AGGREGATION ("Part-of")
  - Building an entity record with descriptors (clustering attributes)
  - COMPOSITION (stronger "Part of" - no independent existence)
- GENERALIZATION/SPECIALIZATION ("Is-a")
  - Forming subtypes/supertypes, population subsets

Collections: (assumes *homogeneous* members)

- SET – no duplicates and no order (the only one in Relational)
- BAG – counting duplicates
- SEQUENCE – order matters

. . .

and looking at Relationships/Structure: GRAPH, TREE, ...

Generalization

SSTYPE

9

see: Graeme Simsion, Ch. 4

- Recognizing commonalities
  - \+ valued - Do I care? Is it useful?
  - – cost - Is it worth the effort?
- Moving "up" to a higher, more inclusive, more generic, more "abstract" view

TYPES:

- Attribute
  - – constrained by Entity Generalization
  - – often a prelude to Entity Generalization
- Entity
  - – represented using subtypes/supertypes
  - – implications for placement and naming of attributes and relationships
- Relationship

Attribute Generalization Examples

SSTYPE

10

Steve Hoberman, *Data Modeler's Workbench*

- For a Tuxedo Rental shop, store Customer attributes:
  - – Waist size
  - – Leg length
  - – Neck size
  - – Arm length
  - – Shoulder width

=> Later add Shoe size.

What does that do to your database schema?

How might you solve the problem?

How does referential integrity become important here?

What is the down side of this schema redesign?

Similarly for Phone numbers:

Problems:

- – Handling international numbers
- – Handling other contact information, e.g. email

N

Attribute Generalization Example

SSTYPE

Steve Hoberman, *Data Modeler's Workbench*

11 Given the following three entity type populations:

What do you observe?

CUSTOMER
- First name
- Last name
- Address line
- City
- State
- Zip code
- Phone number
- Fax number
- Tax id
- First order date
- DUNS #

SUPPLIER
- Company name
- Contact first name
- Contact last name
- Address line
- City
- State
- Zip code
- Phone number
- Cell Phone Number
- Fax number
- Credit Rating
- First PO Date
- DUNS #

ASSOCIATE (Employee)
- First name
- Last name
- Address line
- City
- State
- Zip code
- Phone number
- Pager number
- Social Security #
- Email address
- Hire date
- Clock #

An ASSOCIATE can be assigned to several CUSTOMERs,
and manage the relationship with many SUPPLIERs.
A CUSTOMER or SUPPLIER can contact multiple ASSOCIATES.

N

Attribute Generalization - Financial

SSTYPE

12 Suppose you saw a table defined like this:

How many rows would it have?

What would YOU want to do?

FINANCIAL DATA:

Dept	Year	Qtr	Bud/Act	Category	Amount

Identifier

A Classic Fact Table for a Dimensional Model!

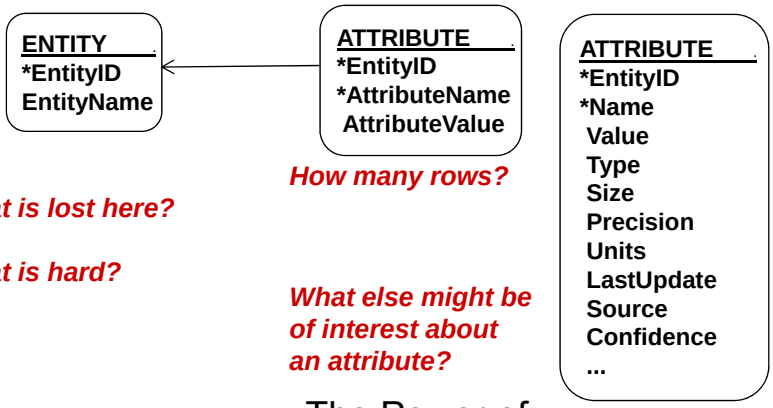
How many rows would this table have?

FINANCIAL DATA:
*Dept
*Year
Qtr1 Budget Material Amount
Qtr2 Budget Material Amount
Qtr3 Budget Material Amount
Qtr4 Budget Material Amount
Qtr1 Budget Labor Amount
Qtr2 Budget Labor Amount
Qtr3 Budget Labor Amount
Qtr4 Budget Labor Amount
Qtr1 Budget Capital Amount
Qtr2 Budget Capital Amount
Qtr3 Budget Capital Amount
Qtr4 Budget Capital Amount
Qtr1 Actual Material Amount
Qtr2 Actual Material Amount
Qtr3 Actual Material Amount
Qtr4 Actual Material Amount
Qtr1 Actual Labor Amount
Qtr2 Actual Labor Amount
Qtr3 Actual Labor Amount
Qtr4 Actual Labor Amount
Qtr1 Actual Capital Amount
Qtr2 Actual Capital Amount
Qtr3 Actual Capital Amount
Qtr4 Actual Capital Amount

Extreme Attribute Generalization

SSTYPE

13



How many rows?

What is lost here?

What is hard?

What else might be of interest about an attribute?

What is easy?

The Power of Generalization Thinking!

If find you are mixing value domains, you may have generalized too much.

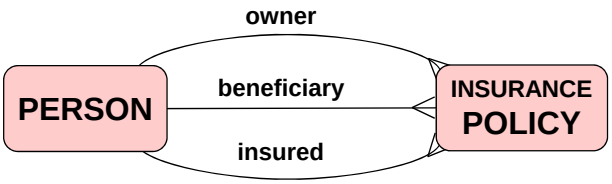
N

Relationship Generalization

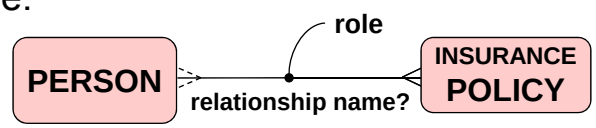
SSTYPE

G. Simsion, *DM Essentials*, 2005, p.140.

14 Reducing multiple relationships:



to one:



NOTE: We have already generalized the individual roles into a Person entity.

Where would you store the 'role' attribute?

How many foreign keys are stored? Where?

N

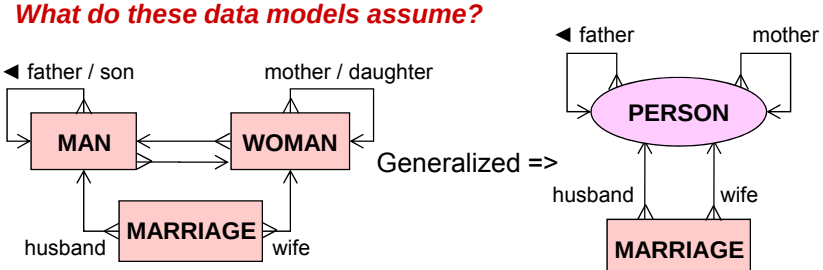
Model "Family Trees"

SSTYPE

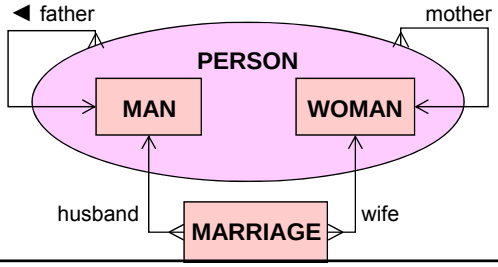
Graeme Simsion, *DM Essentials*, 2005

15 Fathers, Mothers, their Marriages, and Children:

What do these data models assume?



Together with subtypes and supertypes:



... leaving open the question of where the Tables will be built.

N

Generalization: Pros & Cons

SSTYPE

16

+ Fewer entities and relationships (simpler?)
+ Greater flexibility to incorporate extensions
+ Greater long-term stability of the model
+ Looking for commonalities -> greater understanding
+ handle special treatment of subsets

– Hides business vocabulary
  – business terms not in schema names but in attribute values
– harder to express and enforce business rules or constraints. Most become conditional on the specialized subtype.
– Problem defining the identifiers (Ref. modes)
– queries are more difficult to formulate and less efficient to execute (more JOINs).
  – e.g., "FIND Customers WHERE Waist = 42" - no longer works

Entity Generalization

SSTYPE

17

Discerning Inter-Entity Relationships:

PERSON **ORGANIZATION**

EMPLOYEE **SHAREHOLDER**

CUSTOMER **VENDOR**

Construct a high-level, conceptual data model.

Problems?

Observations?

N

A Fundamental Assumption in a Data Model Diagram

SSTYPE

18

- The main construct is an Entity.
- Each labeled box/circle represents an Entity Type
  - a Defined Structure (a schema template)
  - a Population of Instances
- Grouping Instances into Types is essentially Arbitrary.
  - The world isn't naturally that way; the designer imposes a view
- All Entity Type Populations are strictly Disjoint (mutually exclusive; or non-overlapping).
  - At least that is the system's assumption, thus each file/table has its own set of records/tuples.

Is this always true?

What about: **EMPLOYEE** **SHAREHOLDER**

Using Subtypes and Supertypes

SSTYPE

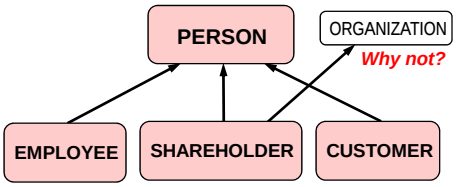
19

Smith & Smith, *ACM TODS*, 1977/6.

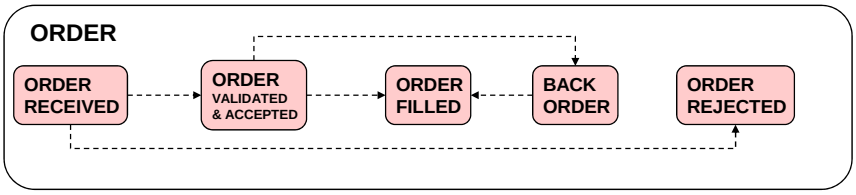
- Subtypes and Supertypes allow us to formally represent overlapping populations
  - Every member of a subtype population *is-a* member of its supertype population(s).

so we can model:

- different *roles* played by members of a common population, e.g.:
  - role determines the attributes



- different *states* of an entity (over time), e.g.:

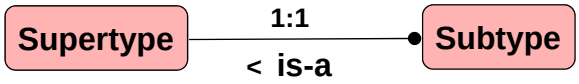


Subtype-Supertype "Relationship"

SSTYPE

20

- Tempting to call it a "Relationship," but …



- pop(subtype) ⊆ pop(supertype)
  SUBSET of
- AND the related members in the two sets are the *same* instance
  - that is what makes it different from relationships in a traditional ER/Relational data model, where the entity type populations are (assumed) disjoint!

Generalization / Specialization

SSTYPE

21 Forming Entity Types

- An "arbitrary" choice made by the Database Designer
- Recognizing when to use Subtypes and Supertypes
- Think about the entity *populations* you are modeling

Two basic and distinct situations:

- Generalization: (bottom-up - from several to a common supertype)
  - When you observe commonalities (e.g., common attributes*) across multiple entity populations.
    - the members may actually be from the same population, the same type of 'thing', so define a common supertype.
- Specialization: (top-down - from one to subtypes)
  - When there is something special about a subset of a population
    - They have some unique attributes*
    - You want to treat them differently
      - e.g., Apply a constraint, or have some attributes mandatory

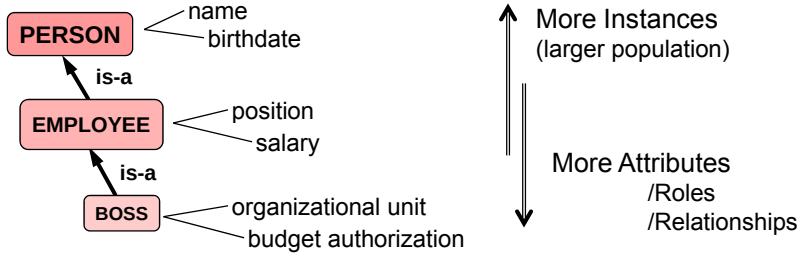
*NOTE: speaking of attributes in ORM, means roles in relationships with other objects.

Subtypes and Supertypes

SSTYPE

22 TWO CONDITIONS MUST *ALWAYS* BE TRUE:

- each subtype population must be a ***subset*** (potentially) of each of its supertype populations
  i.e., each instance of the subtype is in *every* supertype population
- each subtype inherits *all* the roles of its supertypes *and* must have ***additional roles***/relationships



If either condition is NOT true,
no reason to call out the subtype in a separate definition.

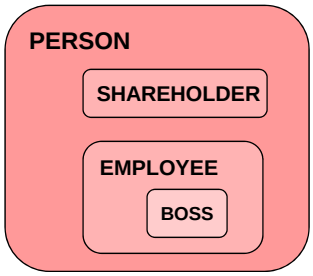
Diagramming Subtypes and Supertypes

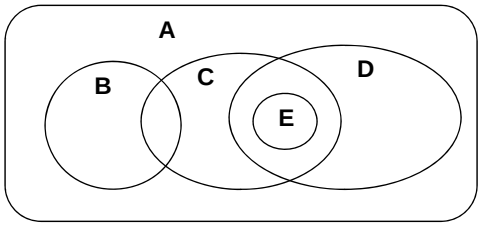
SSTYPE

23

Two Basic Representations:

1. NESTED (Euler Diagram)
   - \+ Intuitive - visually shows inclusion
   - \+ Clean and Compact
   - – Generally assumed disjoint
   - – Only good for simple cases
   - – Not good for complex cases - difficult to represent both exclusive and overlapping subtypes (like a Venn Diagram):





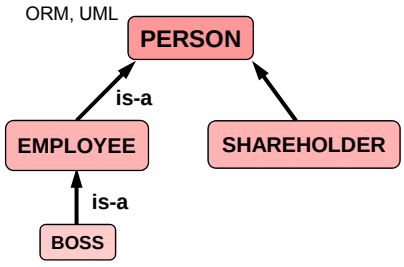
Diagramming Subtypes and Supertypes

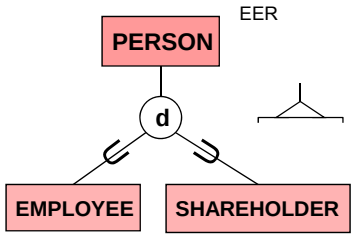
SSTYPE

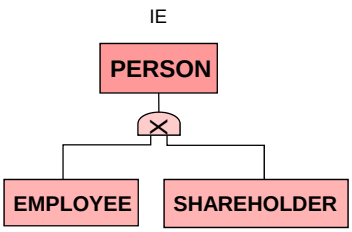
24

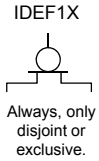
Two Basic Representations:

2. SEPARATED
   - \+ More common
   - \+ Easier to show constraints and multiple supertypes for more complex cases.
   - – not visually intuitive
   - – confusion with "relationship"
   - – Adds more “clutter”









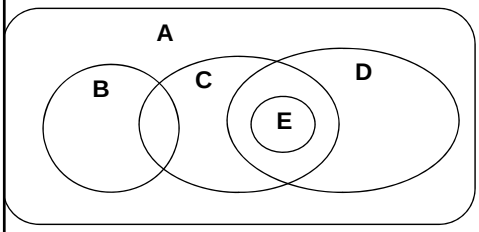
Diagramming Exercise

G

SSTYPE

25

- Convert the following Nested diagram into a Separated S/Stype diagram:



How to model 'E' ?

Any constraints required?

What if only exclusive subtypes allowed?

Subtype / Supertype Constraints

SSTYPE

26

WITHOUT CONSTRAINTS,
ASSUME THE MORE GENERAL CASE:

- overlapping subtype populations
  - else Disjoint, so apply Exclusion constraint:
- non-exhaustive (Partial) on the supertype,
  i.e., a supertype instance need not be in *any* subtype
  - else Mandatory/Totality/Dependency constraint

=> Declare constraints on the more restrictive cases

- Some systems allow only Disjoint and Total
  - it is possible to model Overlapping, even if the system only allows Disjoint subtypes. *HOW*?
- Some systems make Disjoint and Total the defaults

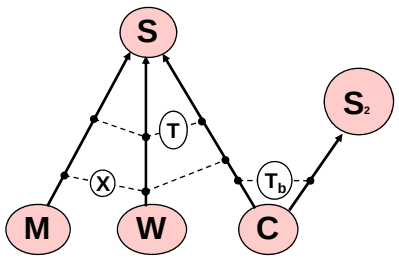
N

Diagramming S/Stype Constraints

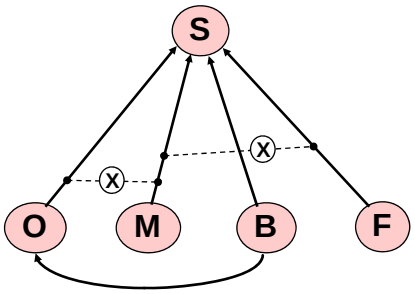
SSTYPE
27

CONSTRAINTS (3 cases):

1. Exclusive or disjoint *subtypes* – X
2. Exhaustive or Total on the *supertype* – T
3. Exhaustive or Total on the *subtype* – T_b



e.g., MAN, WOMAN, CHILD



e.g., OVIPAROUS, MAMMAL, BIRD, FISH

N

S/Stype Constraints - Other Notations

SSTYPE
28

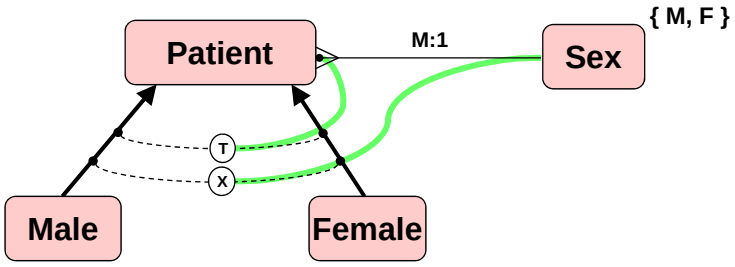
	Exclusive or disjoint	overlapping	Exhaustive or Total	Partial
NIAM	X		T	
ORM*	X	(default)	●	(default)
IE				
EER	d 	O 		
UML ODL	(explicit English labels on the S/Stype arcs)			
IDEF1X	(only)	(can't define)		
*combined				

NOTE: No modeling schemes or systems recognize totality on the subtype.

"Well - Defined" Subtypes

SSTYPE
29

- based on an attribute of the supertype
  - called the "distinguishing" attribute
- characteristics of the relationship determine the constraints on the subtypes
  - mandatory attribute => exhaustive/totality constraint (T)
  - single-valued attribute => exclusive subtypes constraint (X)



- ***What if an optional attribute?***
- ***What if a multi-valued attribute (M:N relationship)?***

Subtype Definition

SSTYPE
30

Attribute-Defined Subtype (Intentional Set)

- a rule for including a Supertype instance in the Subtype
- Defined in terms of the values of a supertype attribute
  - in general, a Boolean expression on attribute(s) of the supertype
- can be considered a *constraint* rule on subset membership
- there are many possible subgroupings (specializations) of an entity type based on the values of its attributes, so find those that matter.
- it is not always possible to define the rules for membership in a subtype, hence: Extensional Set…

Subtype Definition

SSTYPE

31

User-Defined Subtype (Extensional Set)

- inclusion determined by “existence” in the set; membership is manual, the system cannot automate or validate membership.
- Some systems *require* subtype definitions such as *VisioEA* (but... as free-form text!)
- Can always come up with an artificial distinguishing attribute

Fixed Subtype

- Each member instance must belong to that subtype for its whole lifetime, else:

Role or Phase/State Subtype

- An object plays different roles (may be multiple) & can change
- An object moves through various stages or states.

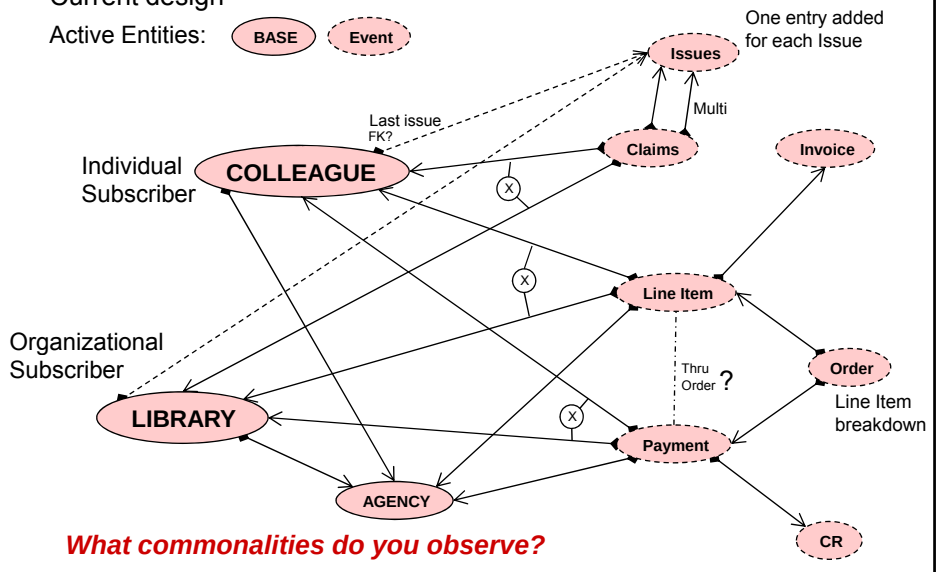
Subscription Database (*MIS Quarterly*)

SSTYPE

32

Current design

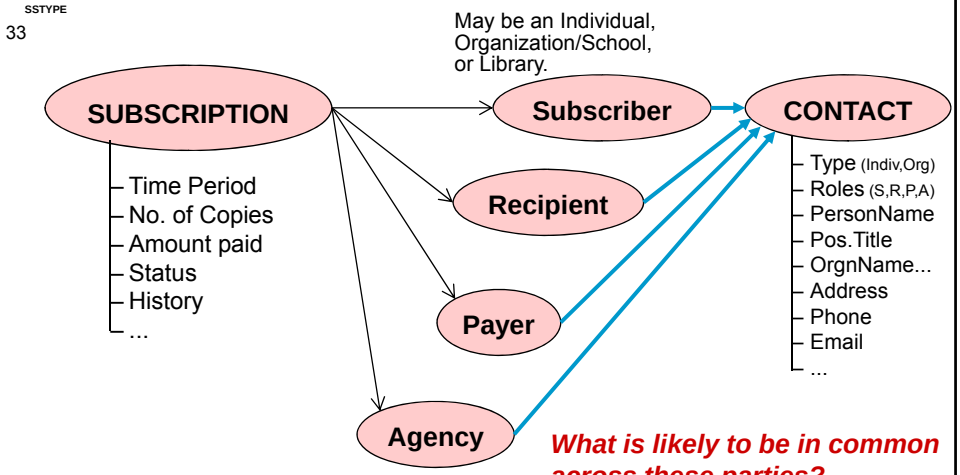
Active Entities: BASE, Event



What commonalities do you observe?

Supertype example - Subscription

SSTYPE
33

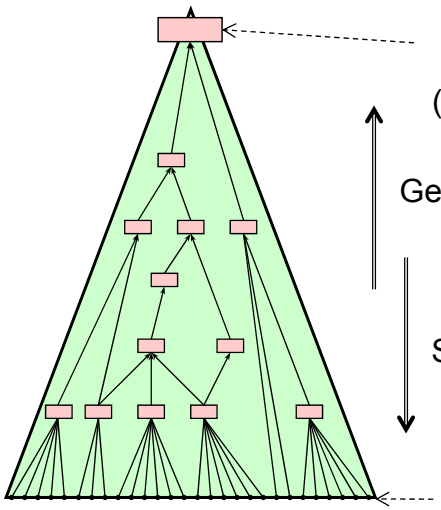


What is likely to be in common across these parties?

Who 'owns' the Subscription? Faculty or the Dept/School that pays?
Gift Subscription? Who gets the renewal notice?

Subtype / Supertype Hierarchy (Lattice)

SSTYPE
34



"UNIVERSAL RELATION"
- A single type population
=> ONE TYPE
(the "root" class in O-O)

Generalization

Specialization

The population of all instances.
- Each instance is its own Type
=> NO TYPES (or N types!)

Sub/SuperType Hierarchy/Lattice - notes

SSTYPE

35

AT THE EXTREMES:

- a single supertype at the top is called the UNIVERSAL RELATION. If you built a single table for all the data in your organization:
  - *what would be the entity?*
  - *what would be the identifier*?
  - *what would be the attributes*?
  - *would all the attributes be relevant for each row*?
- at the bottom would be individual instances, each instance being its own type!
  - But sharing many attributes with other entities

The real art of database design is picking the appropriate entity types *within the levels of the hierarchy*.

- Allows the designer to defer choosing what tables to build

Single vs. Multiple Inheritance

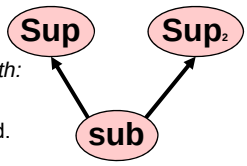
SSTYPE

36

- SINGLE - every subtype has only *one* supertype

  => a strict Hierarchy of types
- MULTIPLE supertypes for a (Shared) Subtype
- If multiple supertypes, they must converge on one population higher up = the root type

  => a "partial lattice" *What's wrong/incomplete with:*



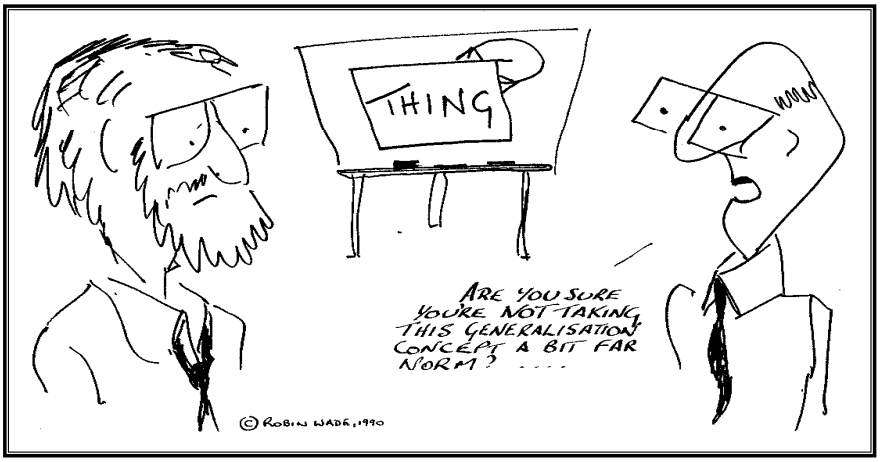
LATTICE = a partially ordered set in which every 2-element subset has both a least upper bound and a greatest lower bound.

- no lattice if no overlapping subtypes
- it is possible to transform multiple inheritance (lattice) to a generalization *hierarchy* by defining all possible combinations of subtypes
- each mini hierarchy has a root (thus partial lattice), all root objects are disjoint.

Extreme Entity Generalization

SSTYPE

37



"The level of generalization is *critical*." - Graeme Simsion

***What is the* 'THING'?**

SIMSION & BOWLES ASSOCIATES

Comparing Modeling Schemes

SSTYPE

38

	ER/Rel (Chen/Codd)	EER (Teorey++)	ORM (Halpin)	UML (OMG)	ODL (ODMG)	SQL 1999
Class Hierarchy	X	Y	Y	Y	Y	Y
Disjoint*		Y	Y	default	only	only
Overlapping		default	default	Y	X	X
Total covering*		Y	Y	user-defined	on abstr.class	partly
Partial		Y	Y	Y (incomplete)	Y	Y
Attribute-defined discriminator		Y	'must' but...	limited (pseudo attrib)	X	X
Shared Subclasses (multiple inheritance)		Y	Y	Y	X	X

*the more restrictive case, calling for a constraint declaration.

Mapping to (Relational) Tables

SSTYPE

39

THREE BASIC CHOICES:

- Supertype only:
  (Absorption - 'flatten' up)

  $\underline{K_P}$ D { P_i }... { A_i }... { B_i }...

- Subtypes only: (not possible in VisioEA)
  (Separation - 'flatten' down)

  $\underline{K_A}$ { P_i }... { A_i }...

  $\underline{K_B}$ { P_i }... { B_i }...

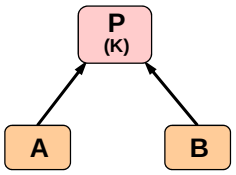
- Both:
  (Partition)

  $\underline{K_P}$ D { P_i }...

  $\underline{K_A}$ { A_i }...

  $\underline{K_B}$ { B_i }...

GIVEN: Halpin$_{08}$, §11.3, p.497.



CONSIDER:
- D = Distinguishing Attribute on P
  (optional)
  (single-valued?)
- Exclusive: on A & B
  - A & B overlapping
- Exhaustive (Total):
  P in neither A or B

PROBLEMS:
- Redundancy
- Incomplete
- Querying

Choosing a Mapping Strategy

SSTYPE

40

MAPPING STRATEGY	Disjoint - nulls	Overlapping	Total - Partial	Queries
on Supertype (absorption) (flatten up)	nulls	multiple type D	+ more nulls	+
on Subtypes (separation) (flatten down)	+	***redundancy***	+ ***arbitrary P*** (no place for P orphans)	joins
on Both (partition)	+	+	+ +	**more joins**

on Sub is best if: #P >> #D

Inheritance and Reuse

SSTYPE

41 • Separate but related notions - often confused (Chris Date got it right)

DESIGN NOTION based on characteristics of populations:

- Multiple populations with some different characteristics sharing some common characteristics,
  … so define a supertype (generalization).
- Need for special treatment of a subset of a population
  … so define a subtype (specialization)
  - Subtype inherits common characteristics from its supertypes plus has some additional characteristics of interest

CONSTRUCTION - implementation efficiency => REUSE

- Inheritance of definitions of data and procedures
  - for efficiency of implementation

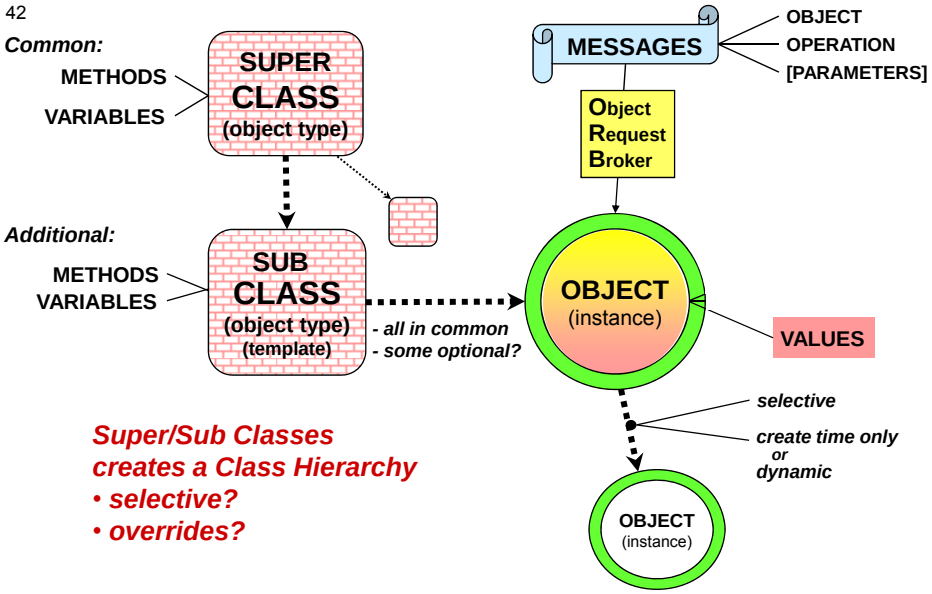
SOLVING PROBLEMS:

- overriding and blocking
- static (copy at creation time only),
  vs. dynamic (maintain linkage to automatically inherit changes)
- multiple inheritance => conflict, priority order

Inheritance - Three Kinds: ⇢

OODBMS

42



Benefits of Subtype/Supertype

SSTYPE

43

G. Simsion, *DM Essentials*, 2005, p.128ff.

- Consciously and Creatively think...
  - commonalities => generalization to supertype
  - special cases => specialization to subtypes
- Greater flexibility to handle extensions.
- Greater stability for long-lived critical applications.
- Generalization can reveal common patterns for reuse.
- Abstraction for presentation, collapse the subtypes into their supertypes
  - equivalent of "leveling" in process/DFD models
- Use subtyping to aid human understanding with no intention of implementing as separate tables.
- Approach design top-down, bottom-up, or middle-out
- Explicit representation of multiple table designs, thus deferring the choice for later implementation.

References

SSTYPE

44

- John SMITH and Diane P. SMITH, "Database Abstractions: Aggregation and Generalization," *ACM TODS*, (2:2) 1977. -classic
- Chris J. DATE, "Subtypes and Supertypes: Setting the Scene," *Database Programming & Design*, 1999 February.
- Terry HALPIN, *Information Modeling and Relational Databases*, Morgan Kaufmann, 2001, in ORM context.
- Graeme C. SIMSION and Graham C. WITT, *Data Modeling Essentials*, 3e, Morgan Kaufmann, 2005, chap. 4.
- Ramez ELMASRI and Shamkant NAVATHE, *Fundamentals of Database Systems*, 3e, Addison-Wesley, 2000, chapter 4 - academic, theoretical.
- Steve HOBERMAN, *Data Modeler's Workbench*, Wiley, 2002, chapter 9 - practical with examples.
- Len SILVERSTON, *The Data Model Resource Book - A Library of Universal Models for all Enterprises*, Vols. 1 & 2, Wiley, 2001.
- Susanne W. DIETRICH and Susan D. URBAN, *An Advanced Course in Database Systems - Beyond Relational Databases*, Prentice Hall, 2005.