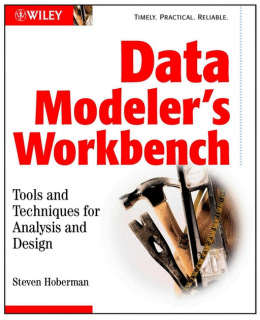
What if you knew the future?

"Wait! I see it now. The winning numbers in tonight's lottery will be…"

Steve Hoberman
me@stevehoberman.com
www.stevehoberman.com

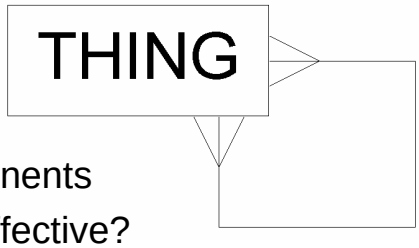
About Steve Hoberman

- Data modeling since 1990
- Instructor for the Data Warehousing Institute
- Author of **The Data Modeler's Workbench**
- Performs 3 roles for organizations
  - Expert Data Modeler
    - Model reviews
    - Tandem designing
  - Data Modeling Instructor
  - Analysis and Design Strategist



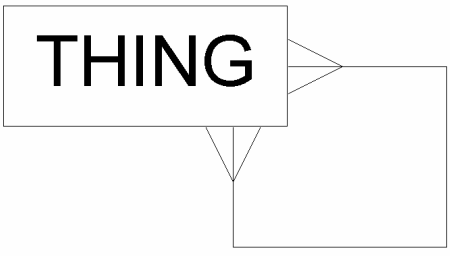
Using abstraction to accommodate future requirements

- What is abstraction?
- Short workshop on applying abstraction
  - Including steps, guidelines & subtyping
- The Good and the Bad
- Abstraction in practice
- What can you abstract?
- Reusable abstraction components
- Where is abstraction most effective?
- Another short (but fun!) workshop
- Other approaches to accommodate future requirements



What is abstraction?

Combining data elements, relationships, and/or entities into a single more generic structure, without denormalizing



Abstraction example - We are given these data elements and...

- 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
- Fax number
- Credit Rating
- First PO Date
- DUNS #

Associate

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

….these business rules

- An Associate can be assigned to many Customers
- A Customer can contact many Associates
- An Associate manages the relationship with many Suppliers
- A Supplier can contact many Associates

Abstracting steps

1. Create a normalized model
2. Abstract where appropriate
   - Data Elements
   - Entities
   - Relationships
3. Document in detail
4. Denormalize where appropriate

1. Create Normalized Model

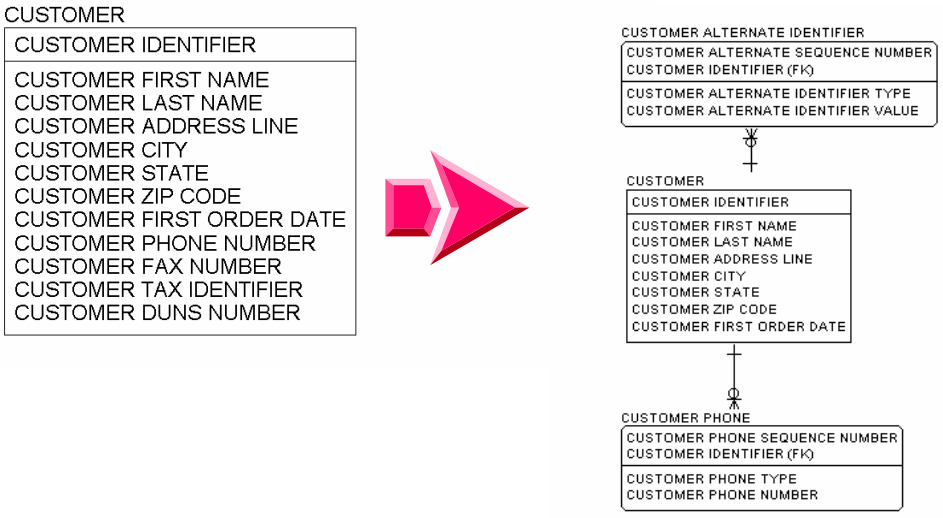
2. Abstract where appropriate

To determine where to abstract, answer these 3 questions:

- Does this data element, entity, or relationship have something in common with one or more other data elements, entities, or relationships?
- If yes, are there concrete situations that would warrant an abstract structure?
- If yes, is the extra development effort now, less than making database changes down the road?

If the answer to all 3 is "Yes!", abstract!

2. Abstract where appropriate - Data Elements

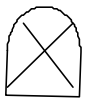


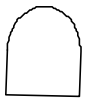
2. Abstract where appropriate - Data Elements

Important Aside - Subtyping Types

	Overlapping	Non-Overlapping
Exhaustive	**Course** **Lecture** **Workshop**	**Associate** **Male** **Female**
Non-Exhaustive	**Person** **Student** **Instructor**	**Contact** **Compliment** **Complaint**

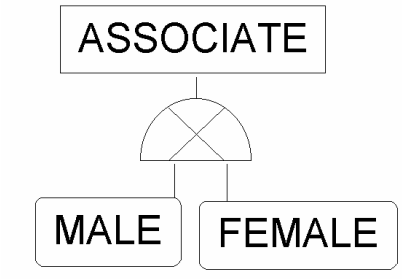
Important Aside - Subtyping Logical Symbols

Non-Overlapping 

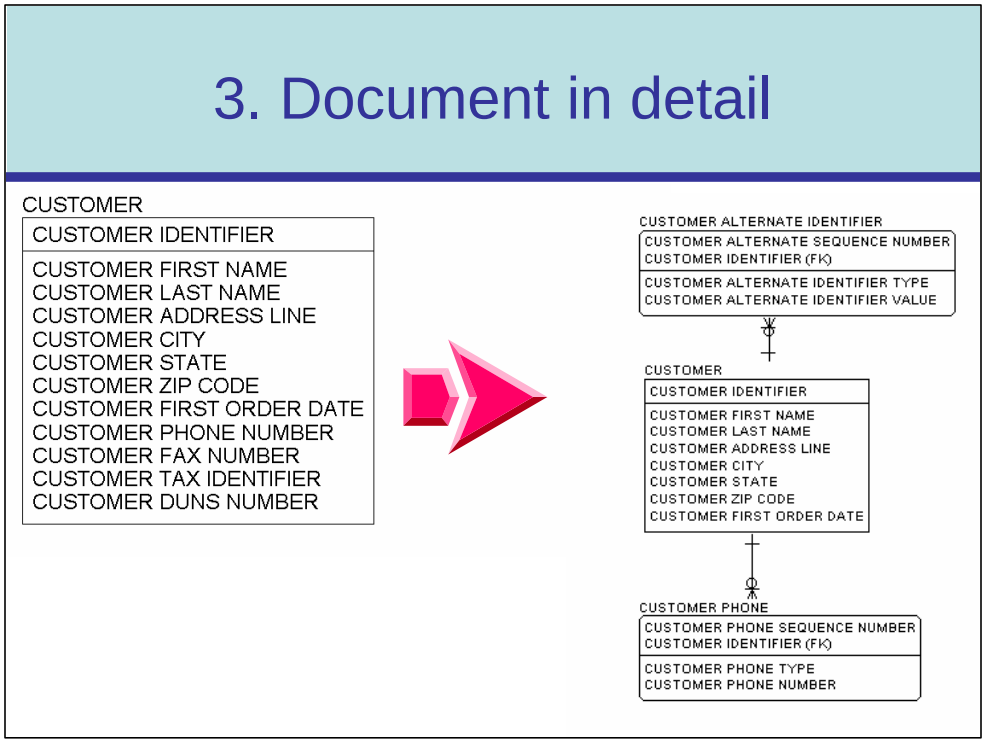
Overlapping 

Exhaustive 

Non-Exhaustive 



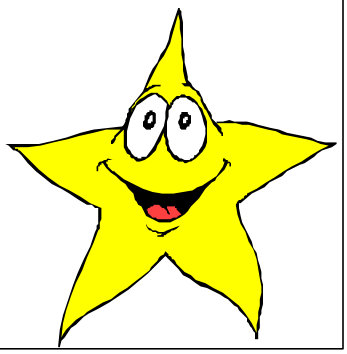
2. Abstract where appropriate - Entities



4. Denormalize where appropriate

The Good

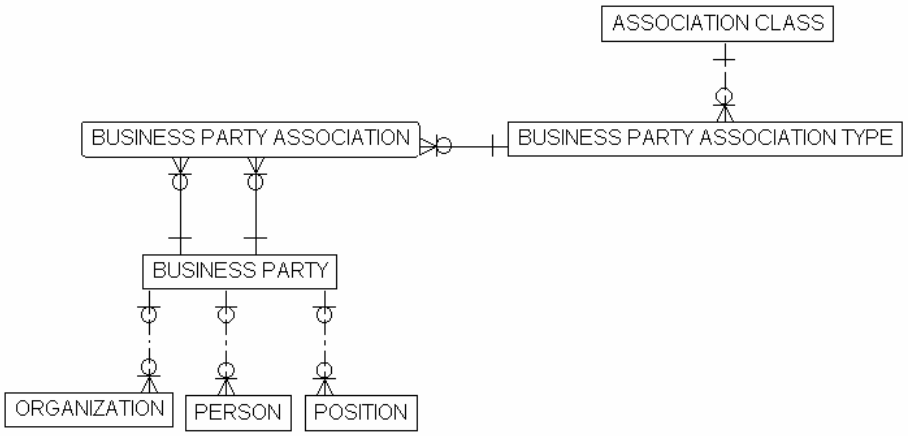
- Flexibility
  - Modeling for the future
- Reduced design time
- Greater understanding of entities
- Less rules in the database?



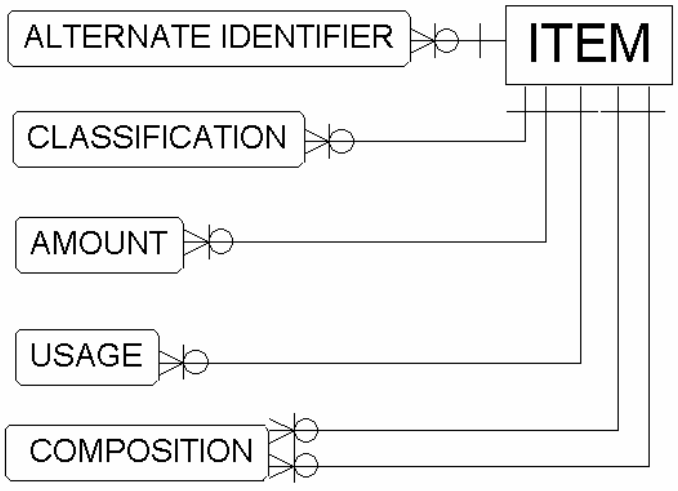
The Bad

- Needs more explanation and documentation
  - "What does a Thing mean?"
  - Meta data needed for row values, not just columns
- Less rules in the database
- Increased development
- Performance issues
- Difficult to justify long term benefits to a project-focused team

Abstraction in Practice - Business Party



Abstraction in Practice - Item

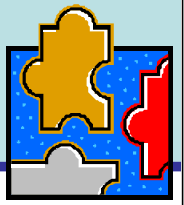


What can you abstract?

- Entities (aka Subtyping)
- Data elements
- Relationships

NOTE: You can dramatically increase the power of abstraction by combining 2 or 3 of these in the same structure

Reusable abstraction components

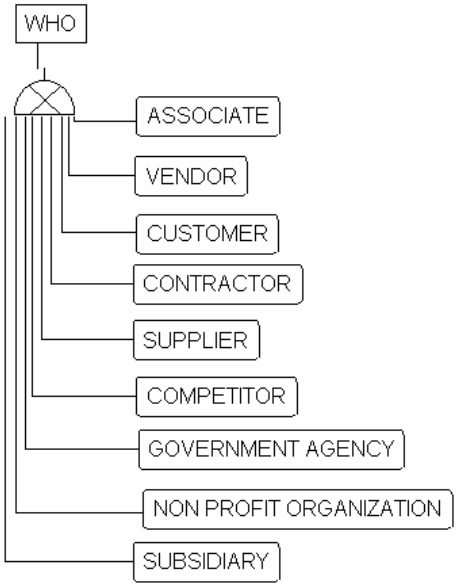


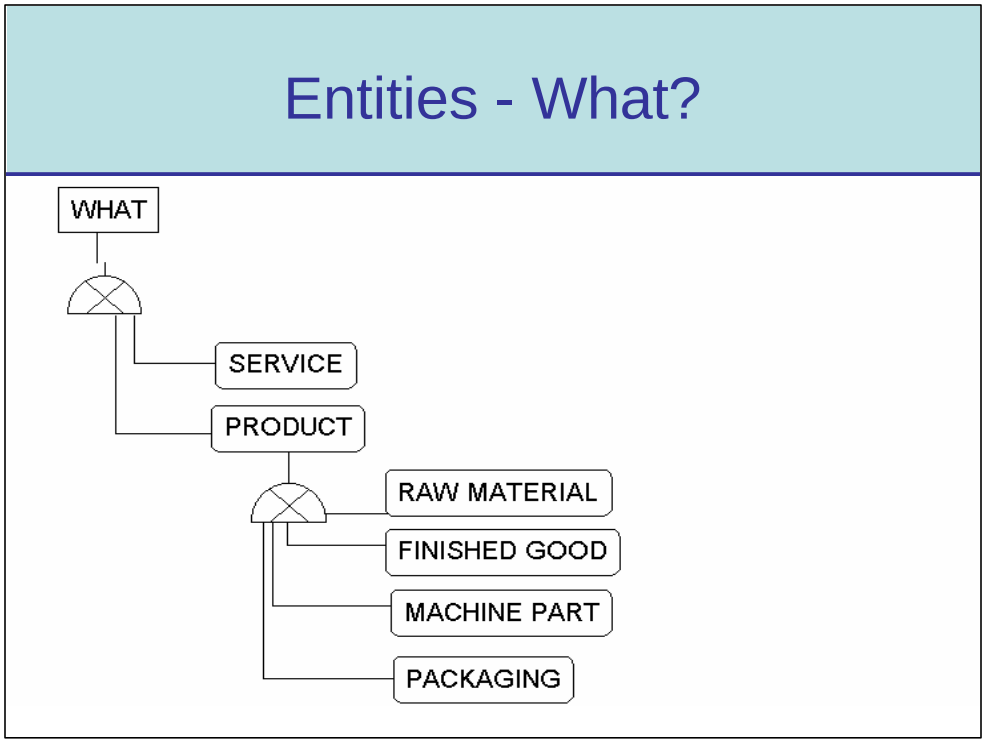
- Pre-built abstract model “pieces” that can replace application-specific sections on the model
- Provides all benefits of abstraction + improves consistency across models
- CAUTION: First understand application before applying these components
- Can create specific to industry or company, but common at high level

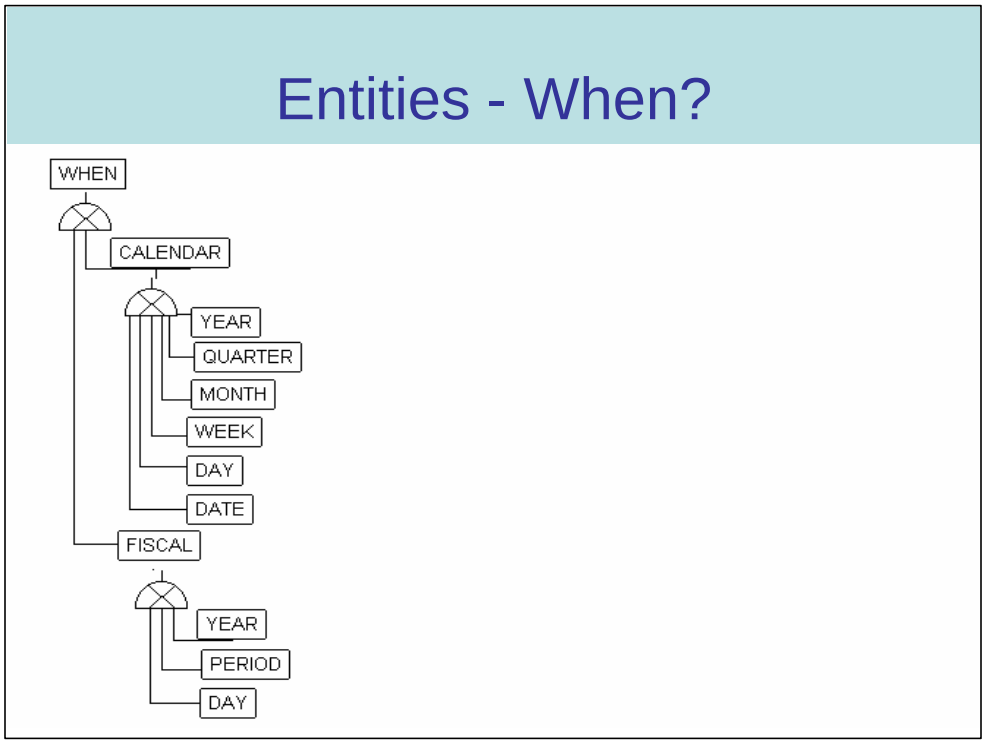
Entities

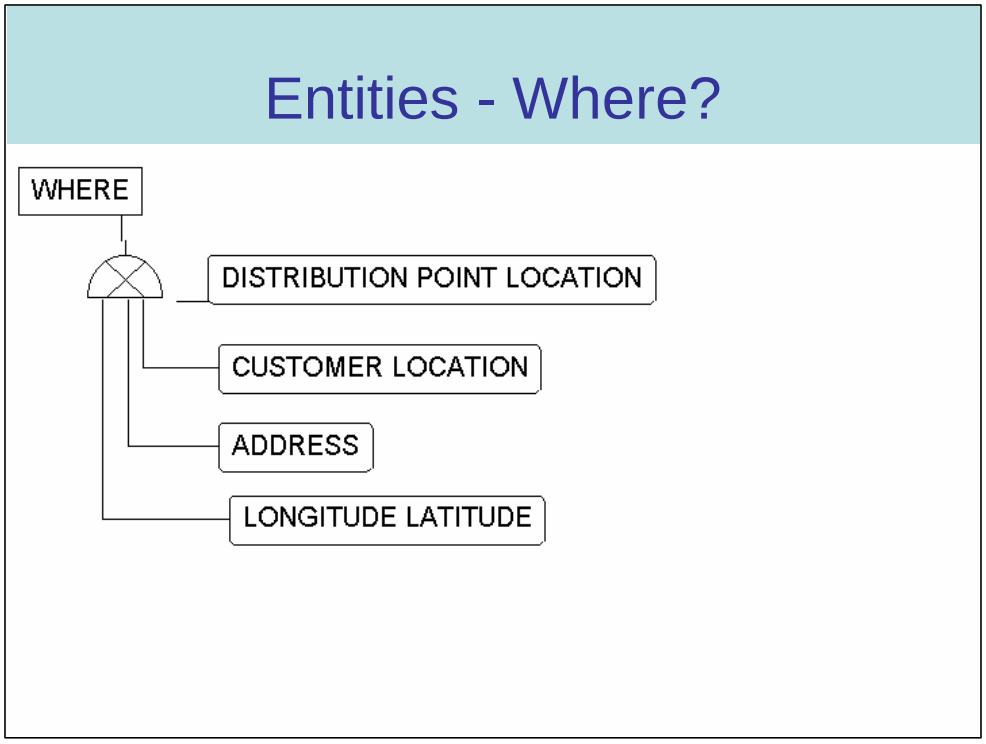
- Who?
- What?
- When?
- Where?
- Why?
- How?

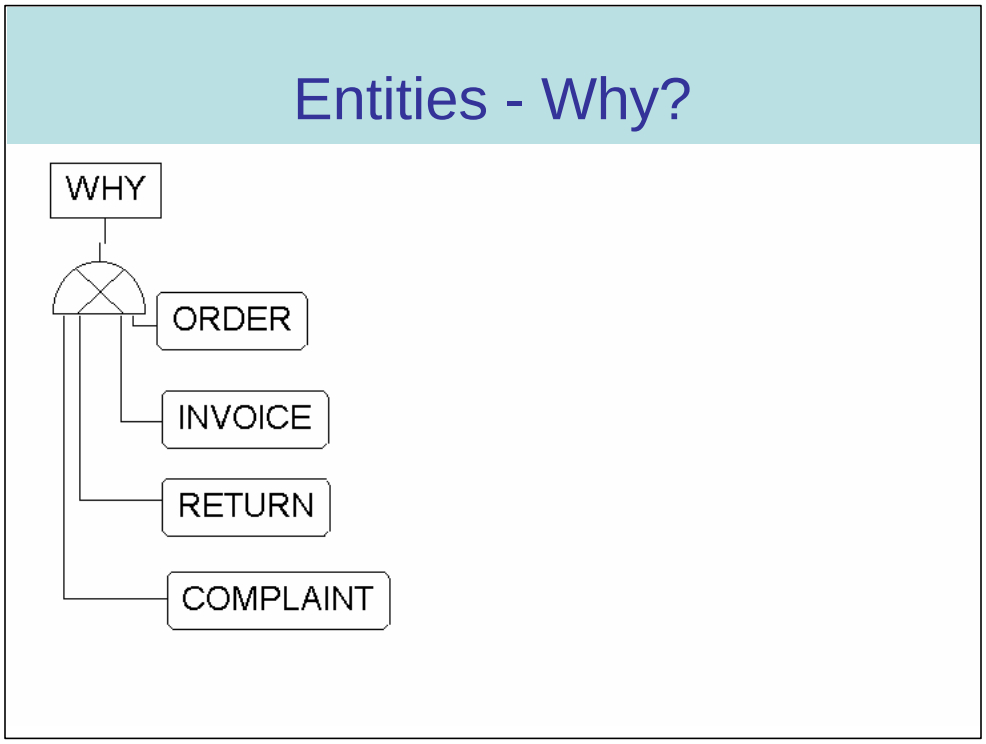
Entities - Who?



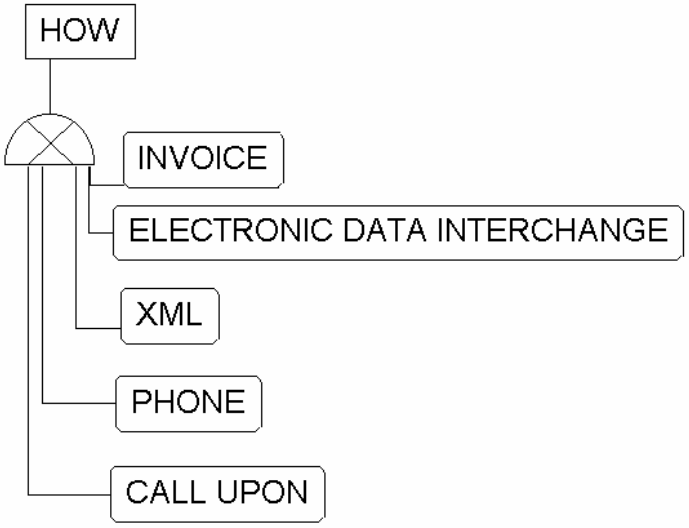






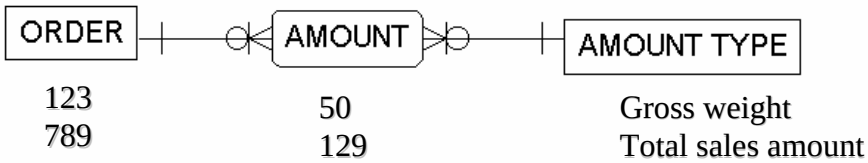


Entities - How?

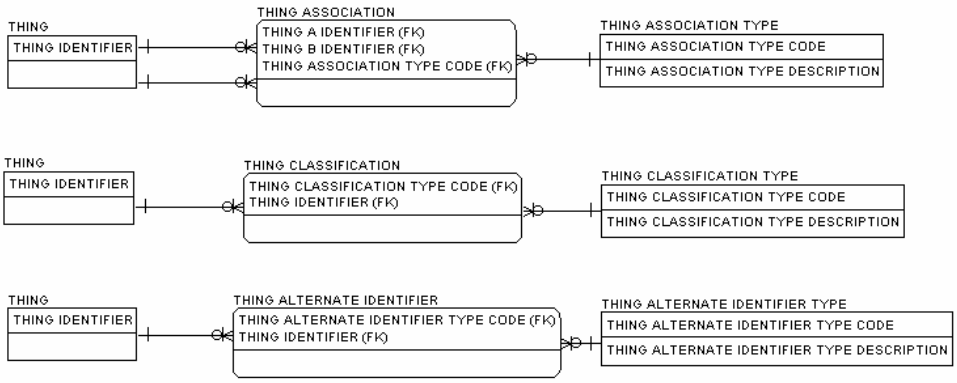


Data Elements

- Highest level is class word:
  - Amount
  - Name
  - Code
  - Date



Relationships



Where is abstraction most effective?

- Designs that require longevity, yet whose requirements can change often
  - Meta data repository
  - Data warehouse
  - Enterprise Data Model
  - Reference databases (Item, Customer)

Create a data model of a meta data repository

Other approaches to accommodate future requirements

- Normalization
- Take more than your immediate needs from source systems
- What else?

Data modeling resources

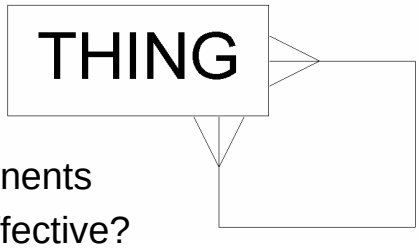
- Books
  - Data Modeler's Workbench, by Hoberman
  - The Data Modeling Handbook, by Reingruber and Gregory
  - Data Modeling Essentials, by Simsion
  - An Introduction to Database Systems, by Date
- Discussion groups
  - www.stevehoberman.com
    - Take the Design Challenge!
  - www.datawarehousing.com
    - Data warehouse topics including modeling, development, tools

Data modeling resources

- Websites
  - www.stevehoberman.com
  - www.dmreview.com
    - Article archive and portal
  - www.infogoal.com/dmc/dmcdmd.htm
    - Portal to data modeling sites
  - www.tdan.com
    - Newsletter on data administration
  - www.EWSolutions.com
  - www infocentric.org
  - www.ralphkimball.com
  - www.billinmon.com
- Email address
  - me@stevehoberman.com

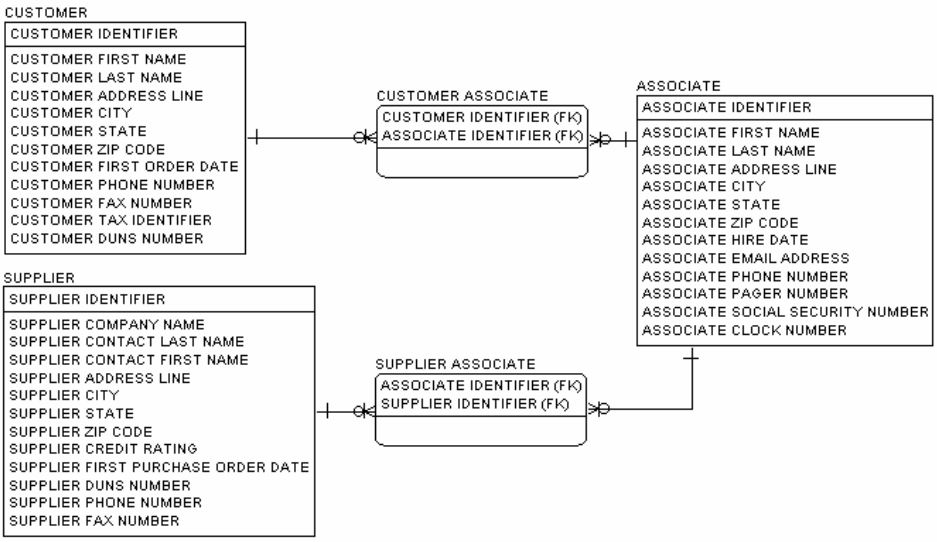
Using abstraction to accommodate future requirements

- What is abstraction?
- Short workshop on applying abstraction
  - Including steps, guidelines & subtyping
- The Good and the Bad
- Abstraction in practice
- What can you abstract?
- Reusable abstraction components
- Where is abstraction most effective?
- Another short (but fun!) workshop
- Other approaches to accommodate future requirements

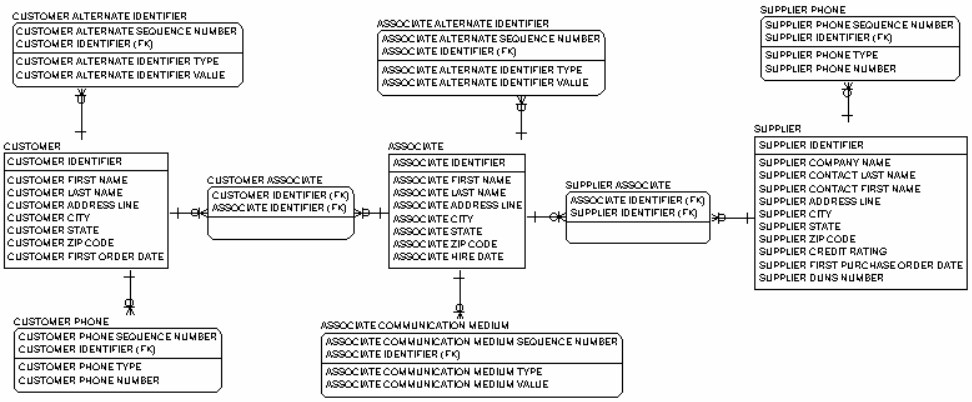


My answers to workshops

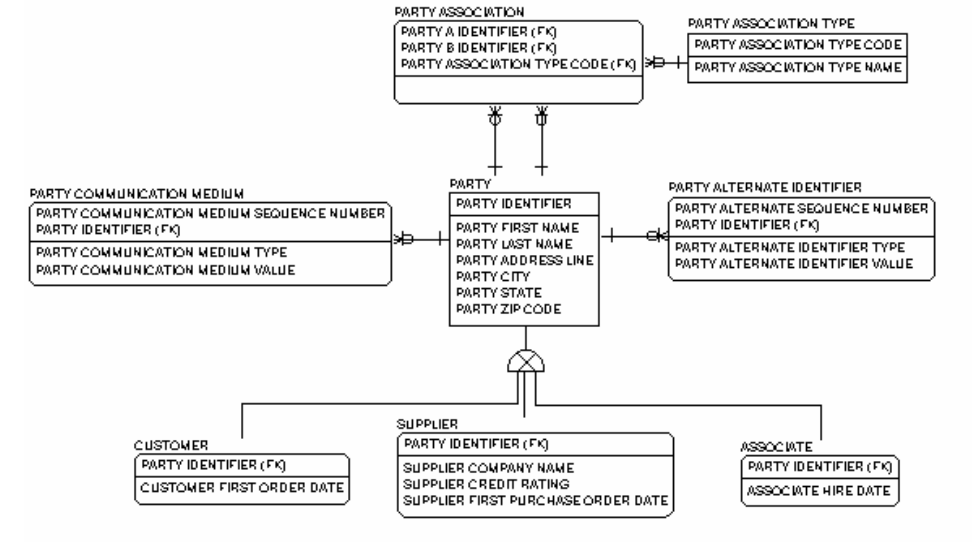
1. Create Normalized Model



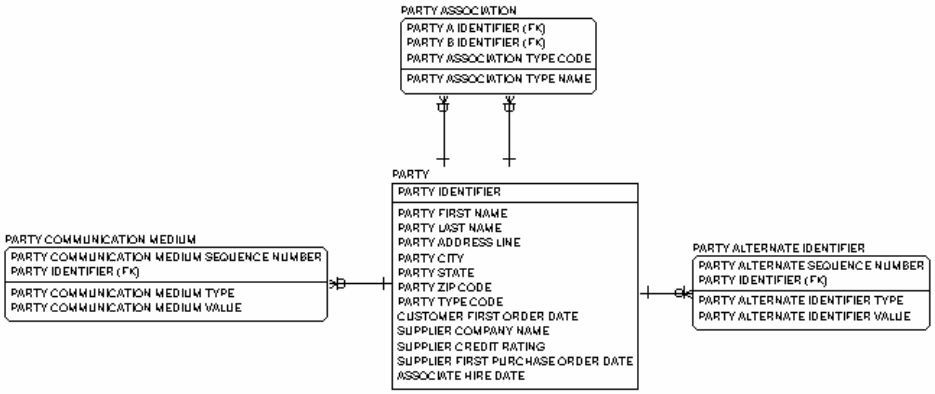
2. Abstract where appropriate - Data Elements



2. Abstract where appropriate - Entities



4. Denormalize where appropriate



Hmmmm….

- Create a data model of a meta data repository
- PS - you have 2 minutes to complete the model, so please use abstraction