suppose this is the data model
bad: not a data model syntactically

- missing relation names
- arrows too small & bad layout
- invented features

Diagram:
- User
- Permanent Shortcut
- Expiring Shortcut
- Shortcut
- URL
- Date
- has

Notes:
- User has Permanent Shortcut and Expiring Shortcut.
- Expired Shortcut has Date.
- URL has arrows too small & bad layout.
bad: not a data model semantically

designer thinking of agents, not atoms

designer is thinking of actions, not state
bad: implementation details

1:1 relation is a tipoff

boolean field: should be subset

collection types not needed

leads to weak multiplicities
bad: wrong semantics

relation not subset; classification is per user

User --> URL --> URLs

User has URL

URL has Permanent and Expiring

Permanent has StartDate

Expiring has EndDate

field names, not sets: disjoint!
bad smells of data modeling
don’t be lazy with names

why?

it’s never as obvious as it seems
choosing a good name helps designer get clarity
beware primitive types

why?

type has syntactic or semantic properties
so may want to store in special way,
and/or use special validators
don't duplicate

why?
recognize shared properties & generalize
leads to cleaner user interface & cleaner code
don’t mention low level ids

why?

every object has an implicit identity anyway
how it’s represented is an implementation detail
but note: user-visible ids (such as usernames) are relevant
don’t split types

why?

distinct types are disjoint, so couldn’t ask whether movie and book have same title
so atoms to be compared must have the same type
don't use set when order matters

why?
  tuples of a relation have no order
  implementer can choose an unordered collection
use subsets, not boolean flags

why?
flag is low level way to represent
obscures dynamic classification
prevents recording multiplicity graphically
don’t use attributes

why?
attributes are an ill-defined idea, and just complicate the notation.
better to factor out the relations that matter
don’t confuse state with actions

why?
data model describes what is *remembered*
that is, what’s stored in the state
don’t use subsets for relational state

why?
subset is with respect to a context (a user)
without this, data structure won’t work
another example of bad subset
collections aren’t domain objects

Why?

Collection objects are implementation details unless they have properties beyond their contents.
don’t split a relation

why?

splitting obscures generalization
leads to duplication in code
watch out for 3-way relations

student-class-grade is a 3-way relation
need a “tuple” type such as Enrollment
another example of 3-way relation
model data that matters

Don’t ask ‘what do I know about students?’

but: ‘what must my app know about students?’

why?
no point modeling the easy stuff
focus on the hard parts
in this case how a student is identified matters
home town probably does not (except maybe for MIT Giving)