http://www.flickr.com/photos/phillipecasablanca/2052686592/

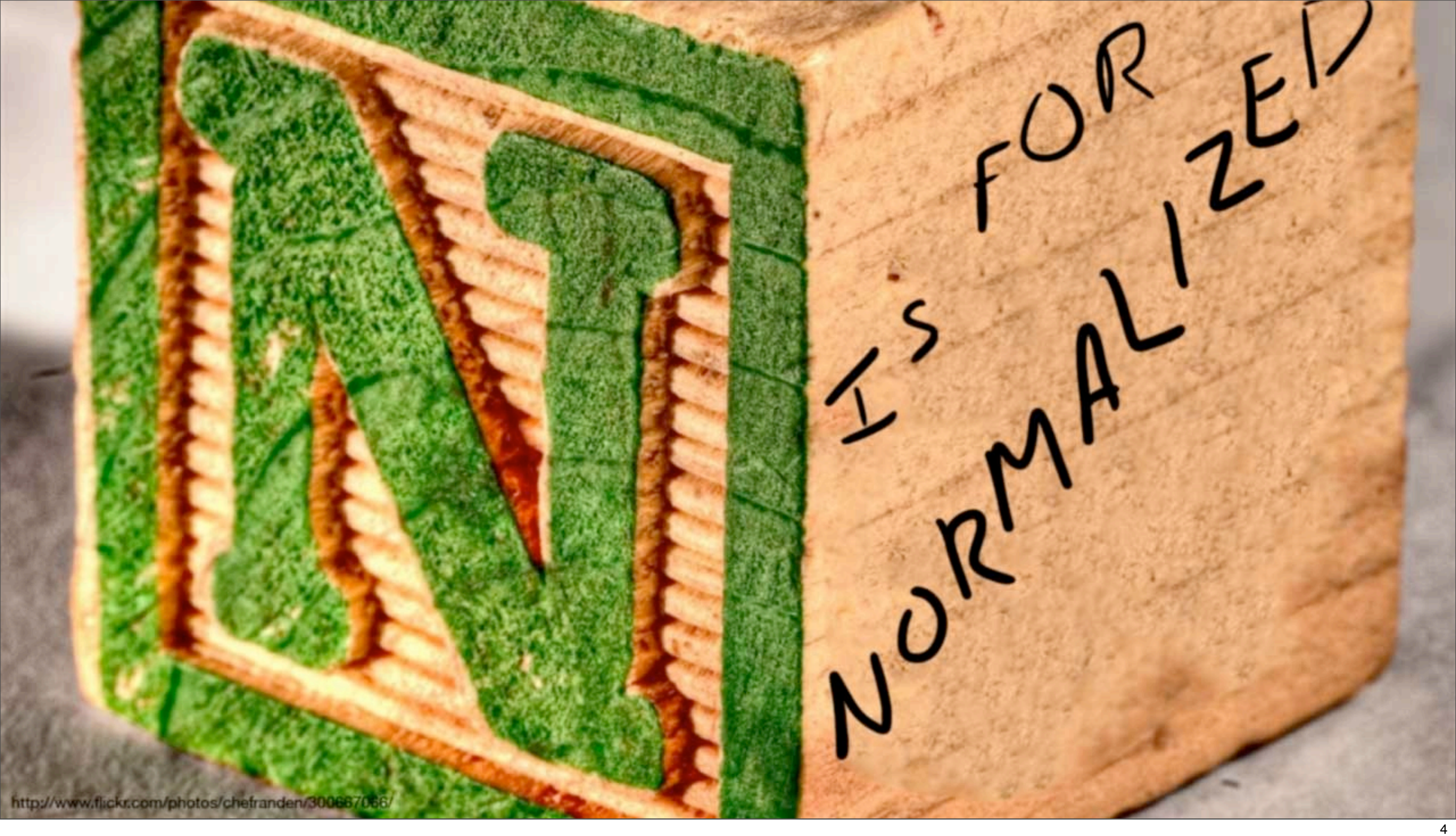
IT'S ALL ABOUT ME

ME ME ME ME ME ME

QUEST SOFTWARE

http://www.flickr.com/photos/bixentro/2091438688/

Before diving into the nasty, muddy, non-relational, confusing details it's important to understand relational theory and how we use it to model data.


K IS FOR
NORMALIZED
http://www.flickr.com/photos/chefranden/300667066/

http://www.flickr.com/photos/jontintinjordan/3736095515/

ALL Attributes are atomic

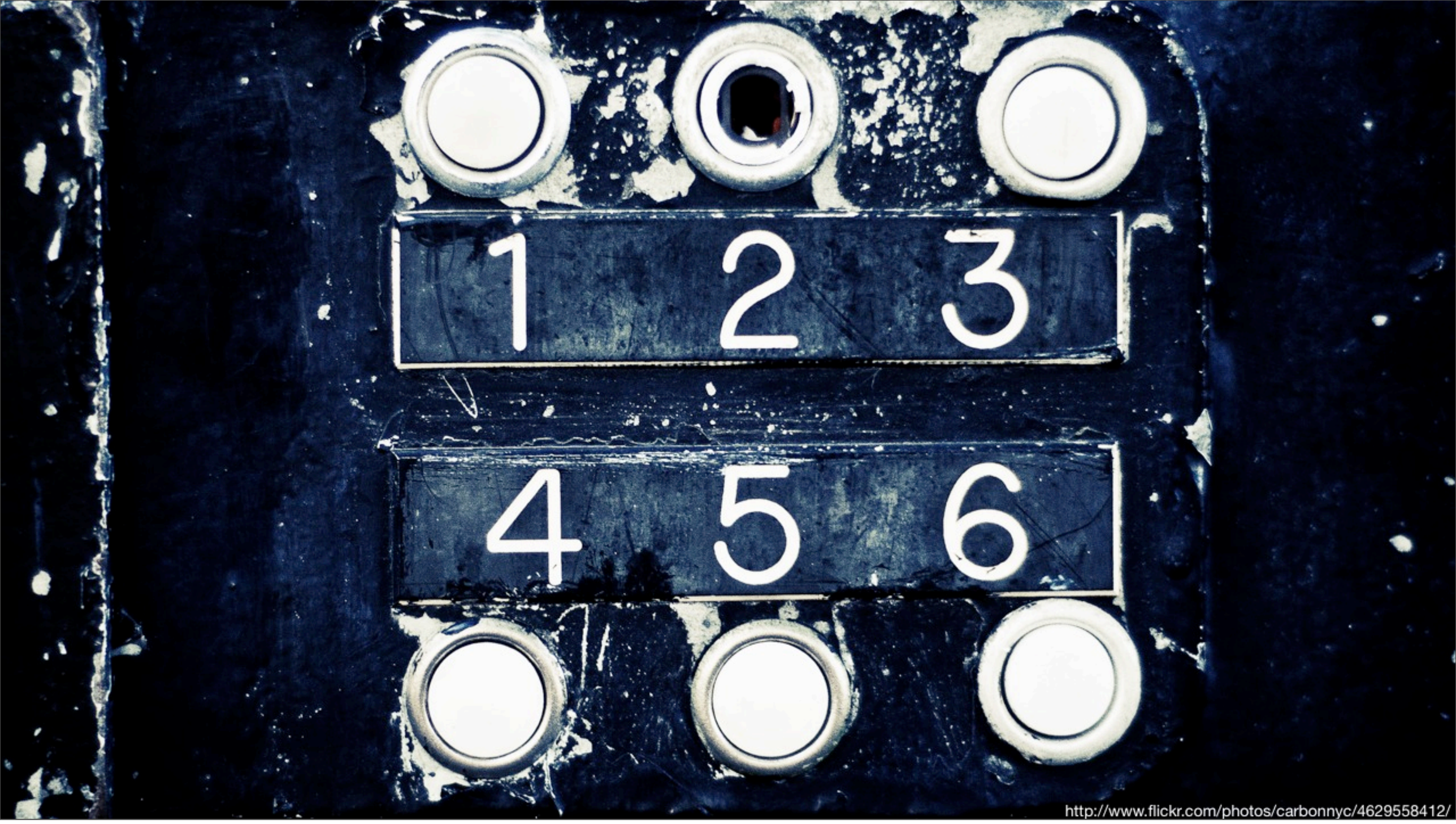
http://www.flickr.com/photos/-cavin-/2313239884/

Columns contain one value

This is tricky: what is a value? Is XML a singular value if it's always consumed as a chunk? What about a binary representation of an object?

http://www.flickr.com/photos/rossaroni/4972841433/

Every row has the same number of columns



http://www.flickr.com/photos/carbonnyc/4629558412/


Every room describes
just one thing

http://www.flickr.com/photos/hokkey/324869104/

columns only describe the key, not other columns can

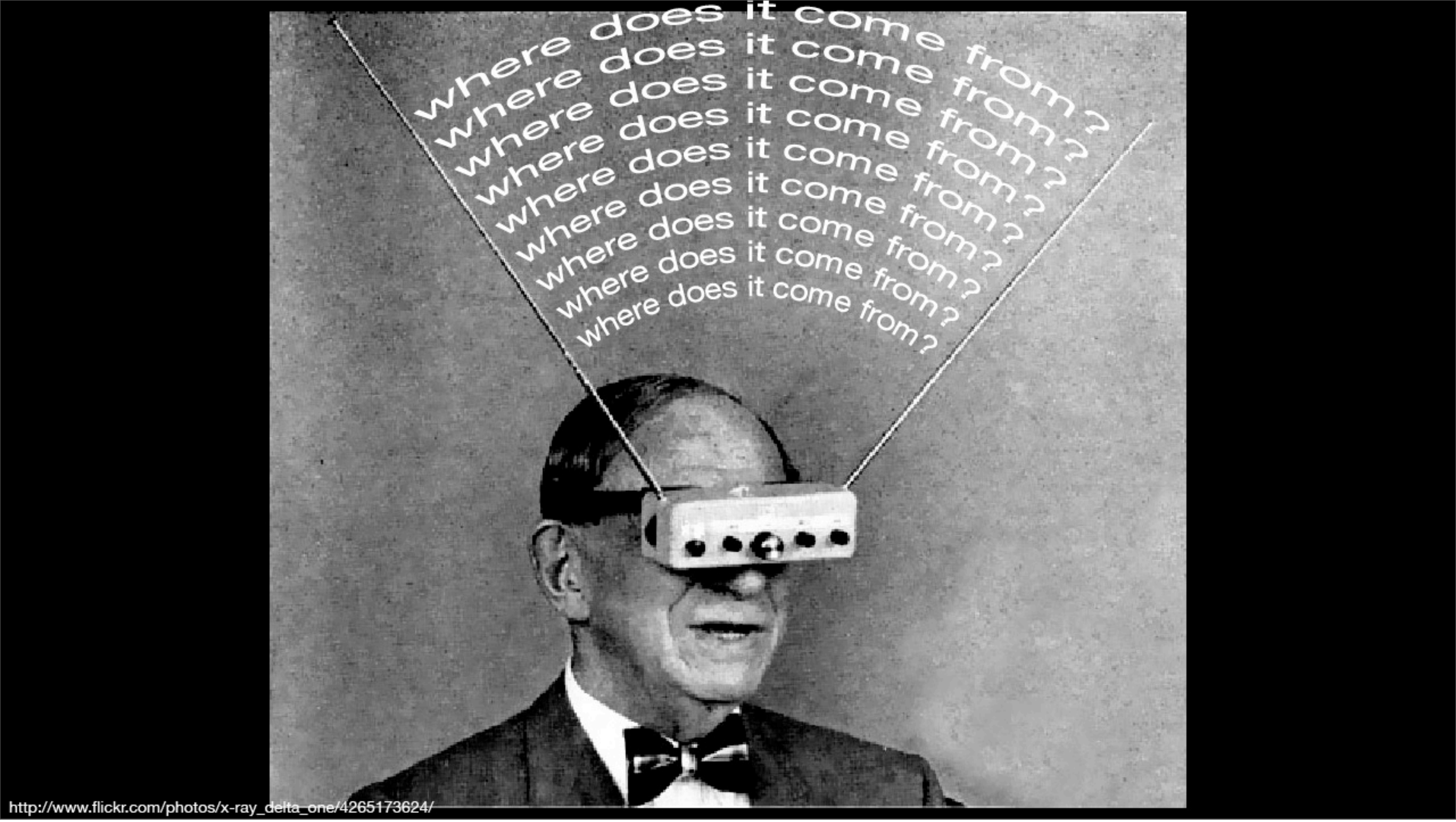
http://www.flickr.com/photos/otherjoel/277955610/

http://www.flickr.com/photos/x-ray_delta_one/4265173382/


Muddy data:
what is it?
http://www.flickr.com/photos/rberteig/195214469/

http://www.flickr.com/photos/dariusdunlap/749610787/

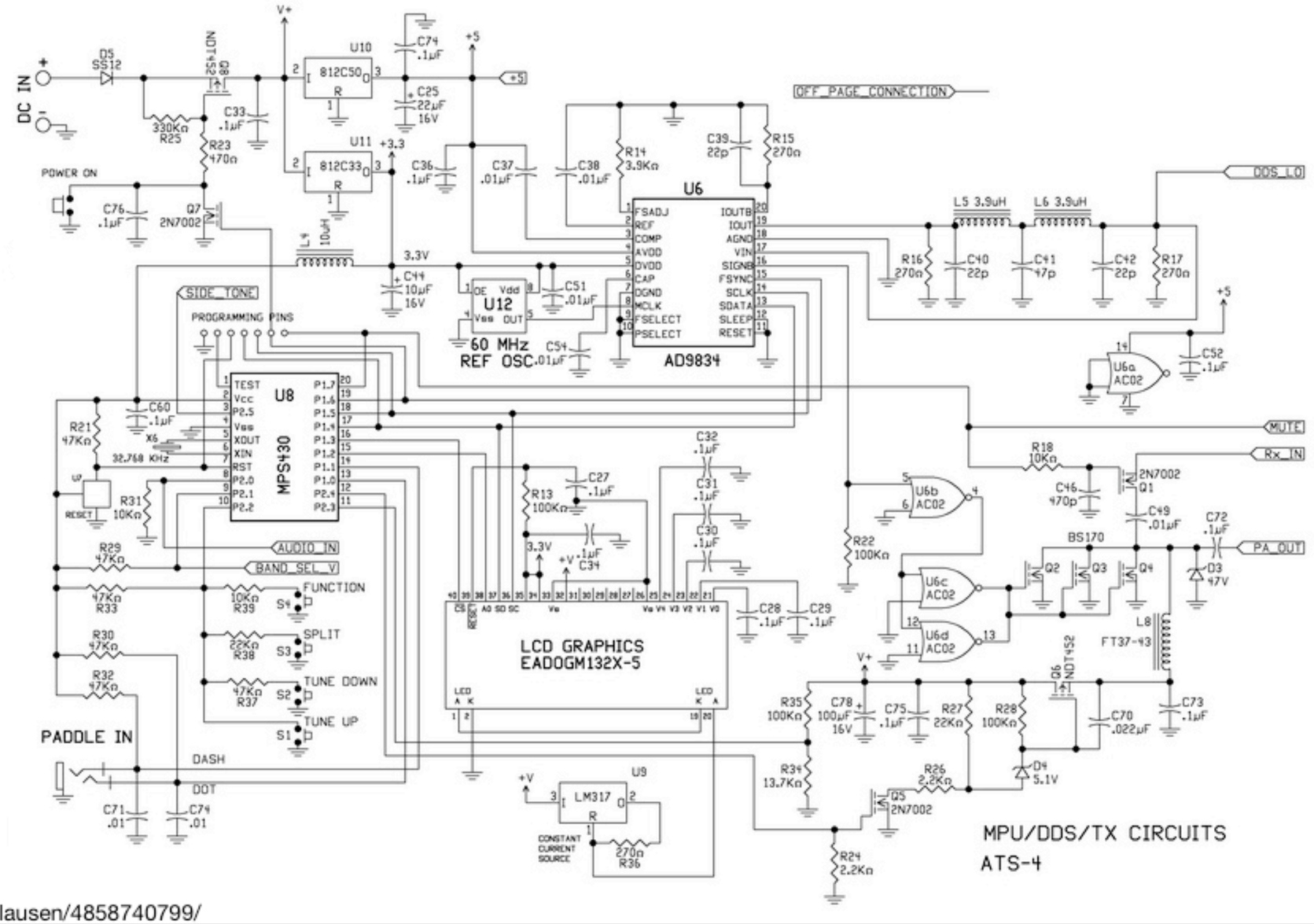
It's data that doesn't fit
Something that isn't easily stored and retrieved from 3NF
Complex attributes
You might call this flexible/dynamic data



Ask questions!

User sessions
Custom data

How would you do it?

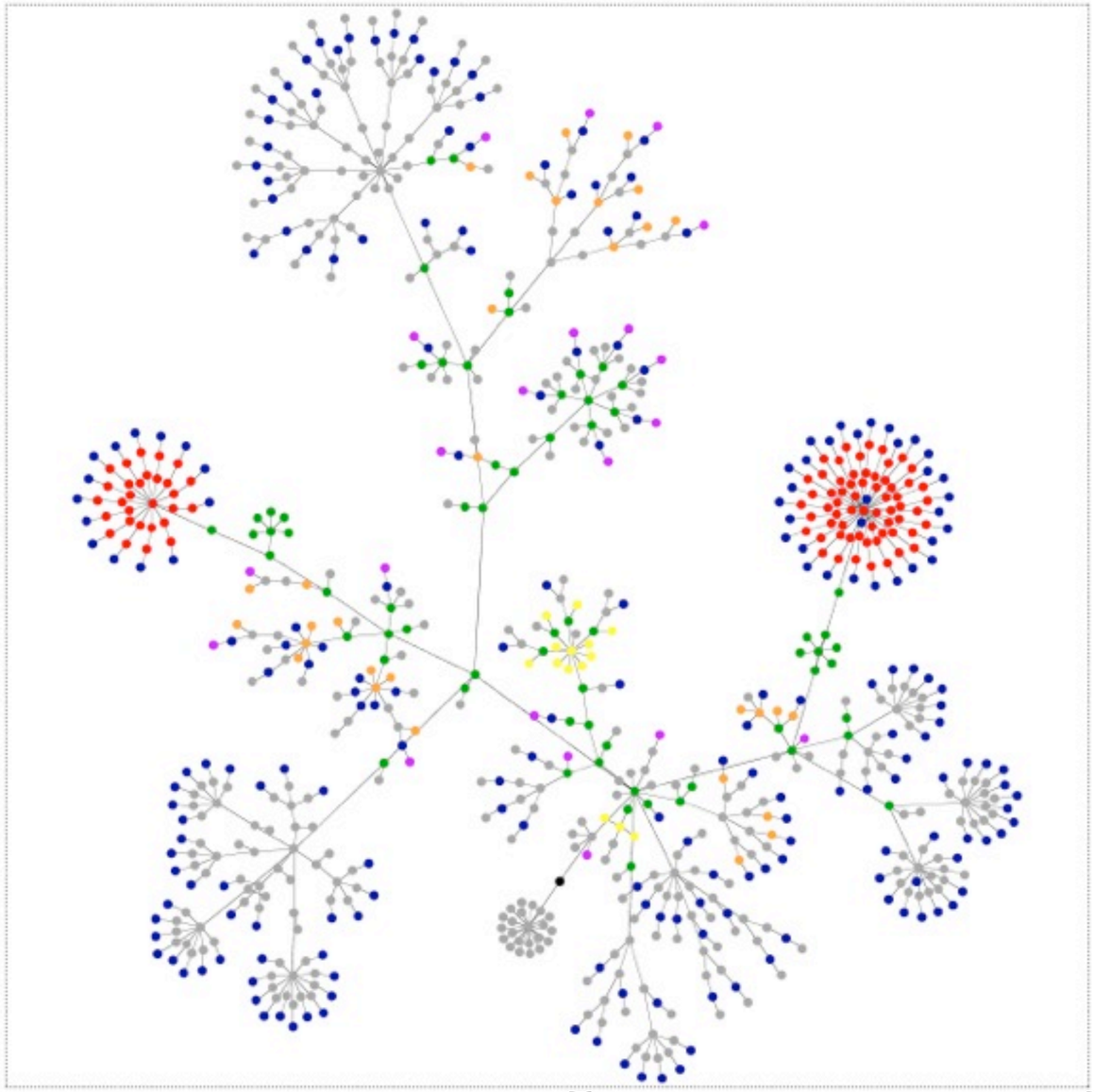


http://www.flickr.com/photos/daveclausen/4858740799/

common solutions

http://www.flickr.com/photos/x-ray_delta_one/3968092988/

NULLable columns



entity:attribute:value

http://www.flickr.com/photos/thefangmonster/352461415/

tracks relevant facts - only record exactly what you need
use this with many individual attributes
requires many descriptor tables to be useable
the physical schema is radically different from the logical schema - impedance mismatch
business logic is in metadata instead of database schema
when to use - sparse attributes, highly dynamic schemas

take a look at VistA database from the Veterans Health Administration for a real world example - lest you think this is a horrible database, it's won a number of awards and is being implemented in other hospitals

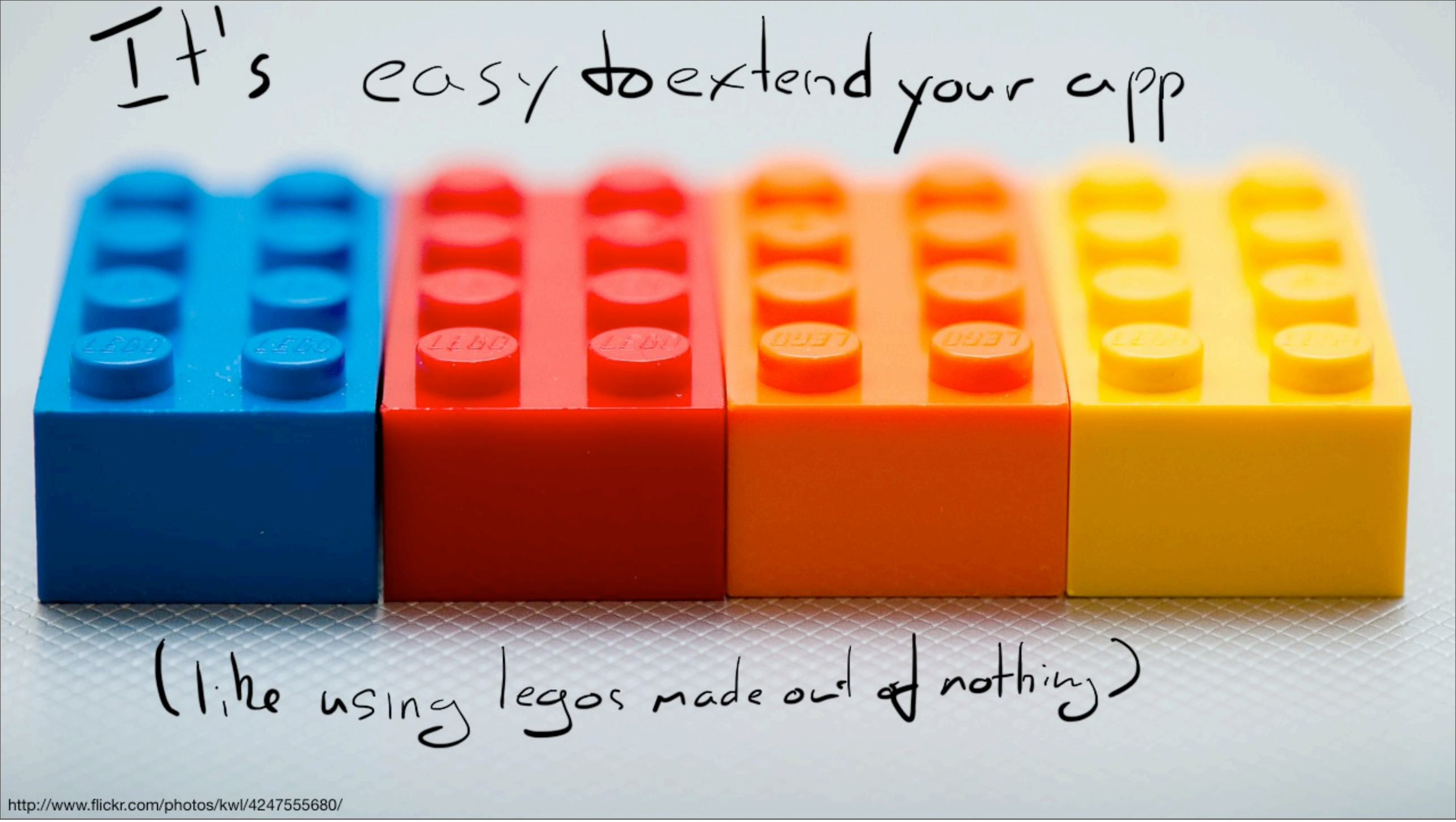
Polymorphic Associations

somewhat specific to OO languages

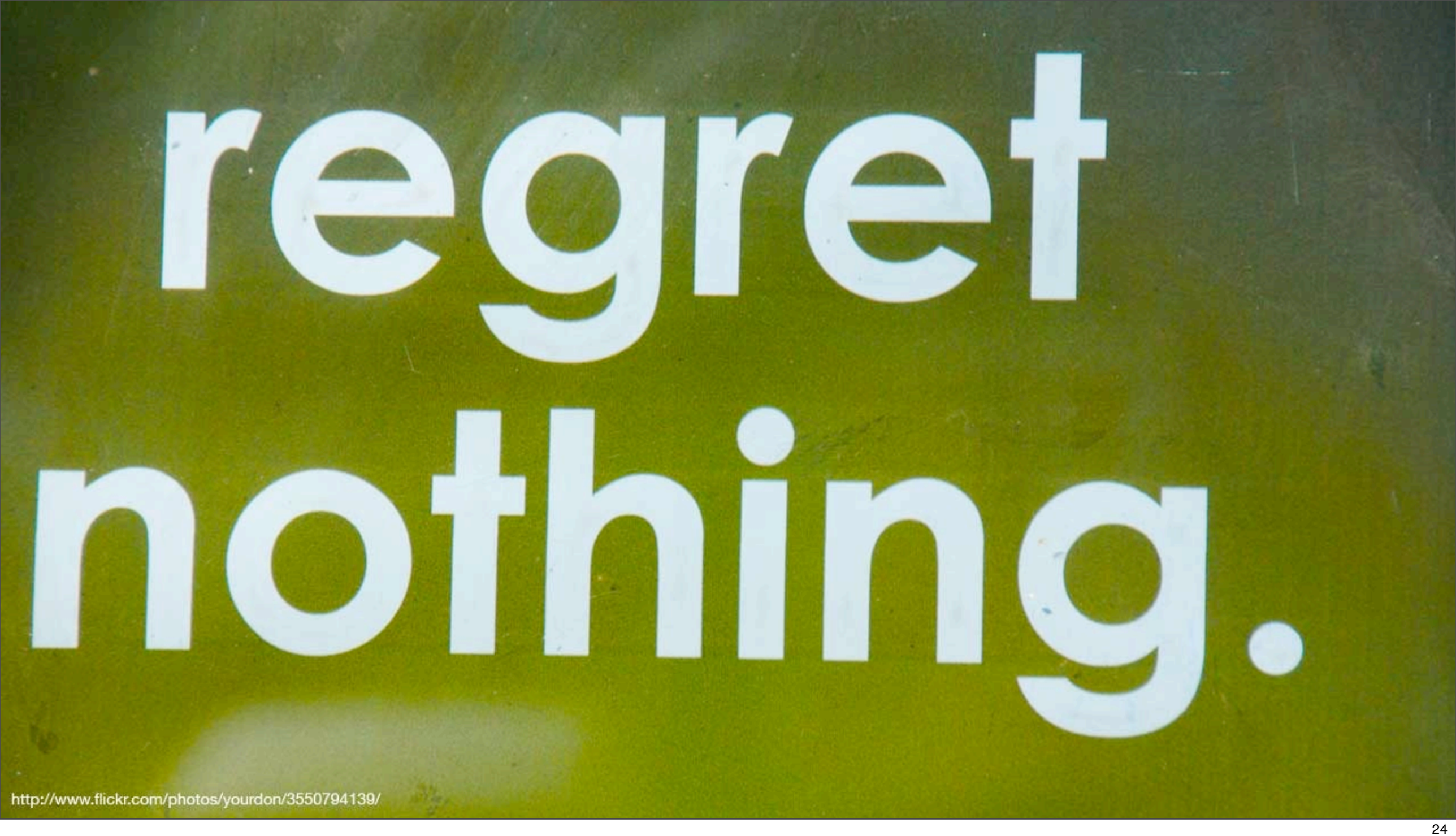
Think of tags - anything can be tagged. Thus, tags need to be generic

Let's talk about NULL.

NULL is an unknown value.
You could argue that optional values are just unknown, right?



http://www.flickr.com/photos/kwl/4247555680/



If NULLs are great, why would we not want to use them?
The purpose of a database is to keep track of data - things we know about. Why bother with things we don't know about?
Becomes difficult to distinguish between NULL and "set to nothing" and we end up with "No data entered" values.
Code becomes complex with checks for NULL values.
Query plans, as a result, become more complex
Three value logic is a hoot! (TRUE, FALSE, and UNKNOWN)

Wide tables are also a PITA to understand, index, and deal with.

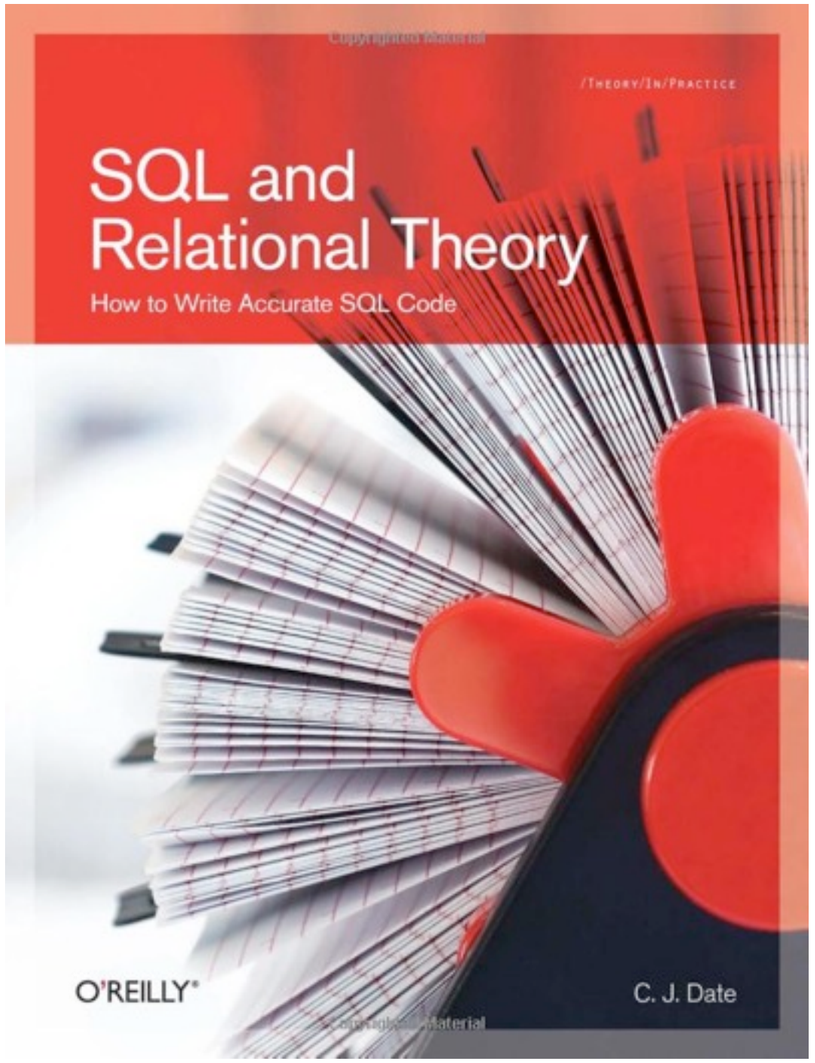
The Academic Reason

Let *T1, T2, ..., Tn* (*n* ≥ 0) be type names, not necessarily all distinct. Associate with each *Ti* a distinct attribute name, *Ai*; each of the *n* attribute-name/type-name combinations that results is an attribute. Associate with each attribute an *attribute value vi* of type *Ti*; each of the *n* attribute/value combinations that results is a component. The set of all *n* components thus defined, *t* say, is a tuple value (or just a tuple for short) over the attributes *A1, A2, ..., An*. The value *n* is the degree of *t*; a tuple of degree one is unary, a tuple of degree two is binary, a tuple of degree three is ternary, ..., and more generally a tuple of degree n is n-ary. The set of all n attributes is the heading of t.

SQL and Relational Theory by C. J. Date. Copyright 2009 C. J. Date, 978-0-596-52306-0.

By definition a tuple (row) contains a value of the appropriate type for each attribute (column). If a NULL is an unknown, it's not a value (NULL != NULL)
Thusly, a table with NULLs isn't really a table at all and is some kind of horrible degenerate monster.

For More Hatred of NULLs...

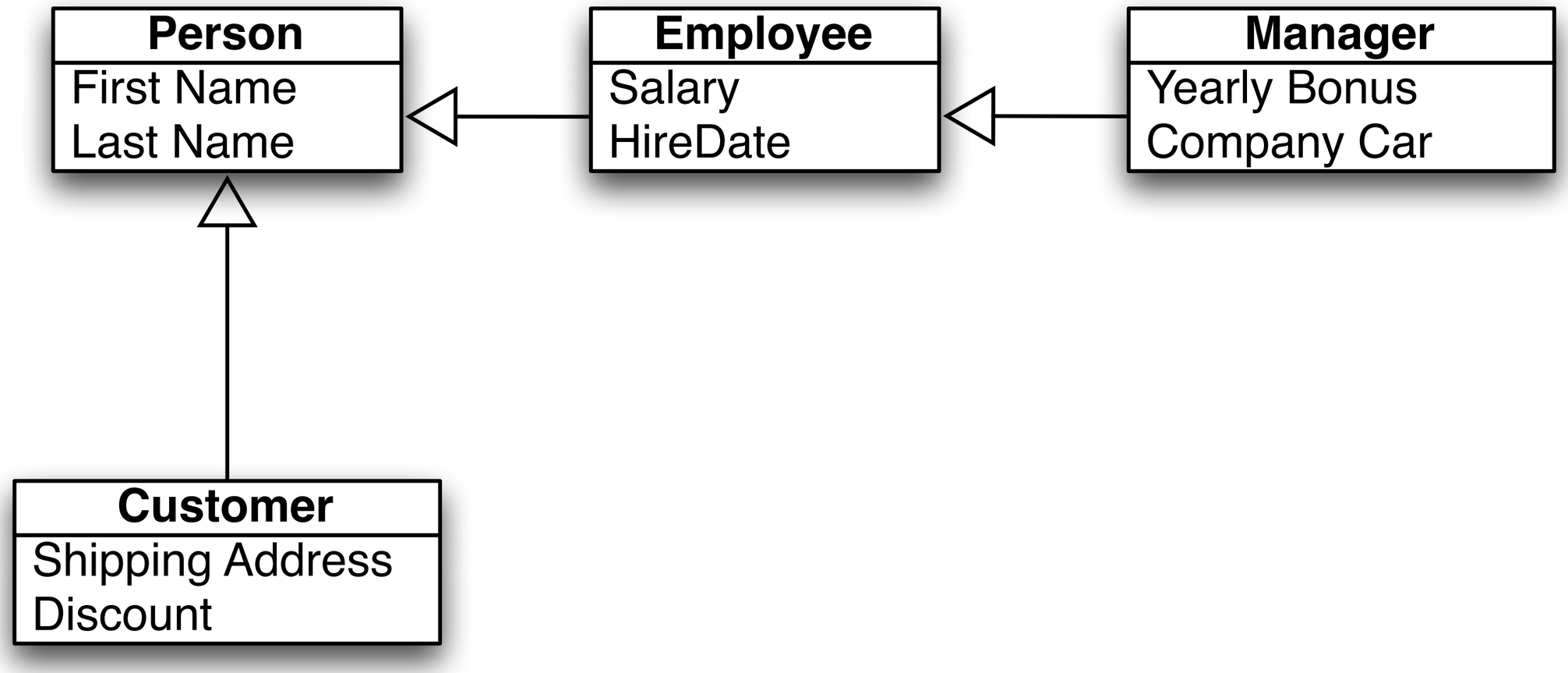


Pros:
Flexible, document structure
Rules can be enforced via a schema
Can be validated both in SQL Server and in the application tier
Can support rich data representations relatively simply

Cons:
Can't use native temporal datatypes in SQL Server 2005 - XML temporal data requires a timezone (you need a workaround)
Breaks the rules of normalization (if you're using NULLs, who cares?)

This is sometimes called 'table per subclass'
Pros:
core data is stored in one table
divide optional data into discrete feature groups then create a separate table for each feature group

Cons:
many 1:1 joins
complex querying
what happens when you sub-class a sub-class?


Person
First Name
Last Name
Employee
Salary
HireDate
Manager
Yearly Bonus
Company Car
Customer
Shipping Address
Discount

Ask yourself questions about how the data will be written and how it will be queried. Once you are comfortable with normalization, usage patterns should dictate the data model, not adherence to "the rules".

Look at frequently run queries, slow queries, indexing patterns, strange/complex joins, tight coupling, cardinalities, access patterns, and shoehorning

there are other ways...

HBase - modelled after Google's BigTable. Fixed column "families", fuzzy columns
MongoDB - document database, has secondary indexes, very much like an RDBMS
CouchDB - document database - flexible data, indexing, stored views, multi-master replication
Redis - in memory key/value store with support for complex data types (lists and sets), new versions support replication, flexible backends
Riak - distributed key/value store with flexible storage backends, distributed, replicated

So why go down this dark and mysterious route? All of these options provide you with varying degrees of flexibility, querying, and scalability. Scaling EAV is hard. Scaling XML querying is hard. Don't be afraid to do something insane.

IMPORTANT INFORMATION

http://github.com/peschkaj/Muddy-Data

http://facility9.com

You should follow me on twitter: http://twitter.com/peschkaj