Rich Modelling

Capturing the context of data as it exists in the wild
Production Tuner for increased oil recovery
State of the art?
Model => Analysis => Decision
Short windows of opportunity
Topside Constraints

Multiple wells & separators all with different characteristics = tradeoffs.
Production Tuner

- near-realtime monitoring
- runs millions of scenarios per day
- not more alarms, best opportunities
- cuts execution lead-time to zero*
- sees hidden edge cases
<p>| Wells | Separators | Export |</p>
<table>
<thead>
<tr>
<th>Key -&gt; Value</th>
<th>Key -&gt; Doc</th>
<th>Column Family</th>
<th>Graph</th>
<th>~Real Time</th>
</tr>
</thead>
<tbody>
<tr>
<td>Riak</td>
<td>MongoDB</td>
<td>Cassandra</td>
<td>Neo4J</td>
<td>Storm</td>
</tr>
<tr>
<td>Redis</td>
<td>CouchDB</td>
<td>HBase</td>
<td>Infinite Graph</td>
<td>Impala</td>
</tr>
<tr>
<td>Memcached DB</td>
<td>Terrastore</td>
<td>Hypertable</td>
<td>OrientDB</td>
<td>Stinger/Tez</td>
</tr>
<tr>
<td>Berkeley DB</td>
<td>OrientDB</td>
<td>Amazon SimpleDB</td>
<td>FlockDB</td>
<td>Drill</td>
</tr>
<tr>
<td>Hamster DB</td>
<td>RavenDB</td>
<td>Accumulo</td>
<td>Gremlin</td>
<td>Solr/Lucene</td>
</tr>
<tr>
<td>Amazon Dynamo</td>
<td>Elasticsearch</td>
<td>HPCC</td>
<td>Titan</td>
<td></td>
</tr>
<tr>
<td>Voldemort</td>
<td></td>
<td>Cloudata</td>
<td></td>
<td></td>
</tr>
<tr>
<td>FoundationDB</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>LevelDB</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Tokyo Cabinet</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>
...there's an 87% chance Linus Torvalds hates your code.

"Bad programmers worry about the code. Good programmers worry about data structures and their relationships."
What about the Relationship Between Sensors?
<table>
<thead>
<tr>
<th></th>
<th>T1</th>
<th>T2</th>
<th>T3</th>
<th>T4</th>
<th>T5</th>
<th>T6</th>
</tr>
</thead>
<tbody>
<tr>
<td>09:00:00</td>
<td>40</td>
<td>41</td>
<td>40</td>
<td>40</td>
<td>67</td>
<td>40</td>
</tr>
<tr>
<td>09:00:01</td>
<td>37</td>
<td>41</td>
<td>40</td>
<td>40</td>
<td>67</td>
<td>40</td>
</tr>
<tr>
<td>09:00:02</td>
<td>40</td>
<td>41</td>
<td>40</td>
<td>40</td>
<td>67</td>
<td>40</td>
</tr>
<tr>
<td>09:00:03</td>
<td>30</td>
<td>41</td>
<td>40</td>
<td>40</td>
<td>67</td>
<td>40</td>
</tr>
</tbody>
</table>
MATCH
WHERE
  rp.distance < 5 AND rn.distance < 5
RETURN
  sp.temp,
  sn.temp
The typical perception of Design Thinking

Holistic Design Thinking
Engineer’s mental model
Match

Pattern
A true story.
h = head switch
i = in
t = tail switch
o = out
x = cross switch
l = low
curr = current switch position
$h =$ head switch
$i =$ in

$hi =$ high
$hol =$ hole
$xil =$ xil
$xol =$ xol
$til =$ til

$t =$ tail switch
$o =$ out

$x =$ cross switch
$l =$ low

$curr =$ current switch position

power

light
h = head switch
i = in

hi

hol xil

x = cross switch
o = out
l = low

curr = current switch position

power

light
h = head switch
i = in

hi_

x = cross switch
o = out
l = low

curr = current switch position

t = tail switch
o = out
“Graph” is a mathematical concept, which was first described in 1736 by Euler. Do not think of pie charts, the picture in your head should look like this:
Legg tallene 1-6 slik at summen av alle sidene er lik.
Det finnes flere løsninger.
**Magic Triangle**

*Directions:* Arrange the numbers 1, 2, 3, 4, 5, 6 so each row has a sum of 9. Use all six numbers and each number once.
Store / Retrieve

Actionable Insights

neo4j
Følg pilene, og lag en vei til kiosken der summen blir 24.
START n=node(*) MATCH n-[r?]-->m DELETE r,m

START n=node(0)
CREATE (A{name:'A'}), (B{name:'B'}), (C{name:'C'}), (D{name:'D'}), (E{name:'E'}), (F{name:'F'}), (G{name:'G'}), (H{name:'H'}), (I{name:'I'}), (J{name:'J'}), (K{name:'K'}), (kiosk{name:'kiosk'}),
  n-[x]-->A, n-[x]-->B, A-[x]-->D, B-[x]-->E, B-[x]-->F, C-[x]-->C, B-[x]-->D, D-[x]-->E, D-[x]-->H, E-[x]-->F, E-[x]-->I, F-[x]-->G, F-[x]-->J, G-[x]-->K, H-[x]-->I, I-[x]-->J, J-[x]-->K, K-[x]-->kiosk,
SET n.name = 'little_guy', A.number = 4, B.number = 2, C.number = 8, D.number = 2, E.number = 6, F.number = 6, G.number = 4, H.number = 2, I.number = 2, J.number = 2, K.number = 4

START n=node(0), kiosk=node:node_auto_index(name = 'kiosk')
MATCH p = n-[a]-->i-[b]-->kiosk
WITH p, SUM(i.number) AS total, COLLECT(i.name) AS name_sequence, COLLECT(i.number) AS number_sequence
WHERE total=24
RETURN p, name_sequence, number_sequence, total
Traverse nodes until there is a 2-way split then follow 3 more segments on both the main branches.
[x IN range(1,10) WHERE x % 2 = 0 | x^x]

[4, 16, 36, 64, 100]
**STEP1: Compute co-occurrence**

Co-occurrence is the basic building block for the algorithm and is in itself a quite useful relationship because it indicates some degree of overlap between tags and therefore a certain degree of similarity which is something that can be exploited for query expansion or recommendation.

The co-occurrence index between two categories A and B is computed as the portion of items in category A that are also in category B, this is a simple division of the number of items tagged as both A and B divided by the number of items in tagged as A.

\[
\text{COOC}(A,B) = \frac{\text{#items tagged as both A and B}}{\text{#items tagged as A}}
\]
what is connected data?

It’s almost as if we didn’t used to know even how to collect data, while preserving its full value. We collected data for a specific set of reports, and when all reports were made, we weren’t able to track details across those reports that came from the same data.

It cost money to store and sift through data. Like an old shirt. If you haven’t worn it in the past year, get rid of it.

Now, we hold the data, and if it’s not stored connected, at least we are capable of reconnecting it.
Questions