Demystifying Big Data: Designing an Architecture for Data and Analytics

March, 2016
Mark Madsen
www.ThirdNature.net
@markmadsen
The complaints about the data warehouse
What is the problem, really? Architecture
Goal, capabilities, organization, implementation
Origin of BI and data warehouse concepts

The general concept of a separate architecture for BI has been around longer, but this paper by Devlin and Murphy is the first formal data warehouse architecture and definition published.

Origins: in 1988 there was only big hair.

- No real commercial email, public internet barely started
- Storage state of the art: 100MB, cost $10,000/GB
- Oracle Applications v1 GL released; SAP goes public, enters US market
- Unix is mostly run by long-haired freaks
- Mobile was this

This is the context: scarcity of data, of system resources, of automated systems outside core financials, of money to pay for storage.
We think of BI as publishing, an old metaphor.

Publishing has value, but may not be actionable.
Planning data strategy means understanding the context of data use so we can build infrastructure.

We need to focus on what people do with information as the primary task, not on the data or the technology.

Monitor → Analyze Exceptions → Analyze Causes → Decide → Act

No problem → No idea → Do nothing
General model for organizational use of data

Monitor → Analyze Exceptions → Analyze Causes → Decide → Act

- No problem
- No idea
- Do nothing

Act within the process
Usually real-time to daily

© Third Nature Inc.
General model for organizational use of data

Collect new data

Monitor ➔ Analyze Exceptions ➔ Analyze Causes ➔ Decide ➔ Act

Act on the process
Usually days/longer timeframe

- No problem
- No idea
- Do nothing
You need to be able to support both paths

Causal analysis, “data science”

Act on the process

Collect new data

Monitor → Analyze Exceptions → Analyze Causes → Decide → Act

Act within the process

Query, reporting, dashboards
The usage models for conventional BI

- Collect new data
- Monitor
- Analyze
- Exceptions
- Analyze
- Causes
- Decide
- Act

- No problem
- No idea
- Do nothing

Act on the process: Usually days/longer timeframe
Act within the process: Usually real-time to daily

This is what we’ve been doing with BI so far: static reporting, dashboards, ad-hoc query, OLAP
The usage models for analytics and “big data”

- Collect new data
- Monitor
- Analyze
- Exceptions
- Analyze Causes
- Decide
- Act

- Act on the process
  - Usually days/longer timeframe

- No problem
- No idea
- Do nothing

- Act within the process
  - Usually real-time to daily

Analytic and big data is focused on new use cases: deeper analysis, causes, prediction, optimizing decisions.

This isn’t ad-hoc, reporting, or OLAP.
Technology captures observations. These change our understanding. New understanding changes practices. Practices drive changes to technology, capturing more data.
Where’s my data?

I never said the “E” in EDW meant “everything”…
It’s going to get a lot worse

Conclusion: any methodology built on the premise that you must know and model all the data first is untenable
Old market says: There’s nothing wrong with what you have, just keep buying new products from us.
The emerging big data market has an answer...
The data lake
The data lake after a little while
“Big data is unprecedented.”

- Anyone involved with big data in even the most barely perceptible way
Big Data Analytics History Lesson: In the 1980’s, the CPG / Retail industry transition from bi-monthly audit data to scanner data changed the dynamics of the industry.

- In late 1980’s, POS scanner data replaced bi-monthly audit data.
- Data volumes jumped necessitating next generation of platforms and analytic tools.
- Leading companies exploited new data and technologies for competitive advantage.

Competitive Advantage:
- Demand-based Forecasting
- Supply Chain optimization
- Trade Promotion Effectiveness
- Market Basket Analysis
- Category Management and Merchandising
- Price Optimization and Merchandise Markdown
- Customer Loyalty Programs

Source: Bill Schmarzo, EMC
Orders of magnitude: 20 years ago TB, today PB

Shifts in data availability by orders of magnitude necessitate new means of managing and using it.
“Big” is well supported by databases now

Source: Noumenal, Inc.
Analytics embiggens the data volume problem

Many of the processing problems are $O(n^2)$ or worse, so moderate data can be a problem for DB-based platforms.
Much of the big data value comes from analytics

BI is a retrieval problem, not a computational problem.

Five basic things you can do with analytics

▪ Prediction – what is most likely to happen?
▪ Estimation – what’s the future value of a variable?
▪ Description – what relationships exist in the data?
▪ Simulation – what could happen?
▪ Prescription – what should you do?
What makes data “big”?

Very large amounts
Hierarchical structures
Nested structures
Linked structures
Encoded values
Non-standard (for a database) types
Deep structure
Human authored text

“big” is better off being defined as “complex” or “hard to manage”
Categorizing the measurement data we collect

The *convenient* data is the transactional data.

- Goes in the DW and is used, even if it isn’t the right measurement.

The *inconvenient* data is observational data.

- It’s not neat, clean, or designed into most systems of operation.

The *difficult and misleading* data is declarative data.

- What people say and what they do require ground truth.

*We need an architecture that supports all three categories.*
Transactions vs “big data”

The classic example of “structured data”

Transaction data includes:

- quantification details (date, value, count)
- reference data for explanation (product, customer, account)
- Lots of meaningful information

Reference data is usually shared across the organization, hence its importance. There are two parts:

- identifier to uniquely identify the subject
- descriptive attributes with common or standardized value domains
Today it’s different data: observations, not transactions

Sensor data doesn’t fit well with current methods of collection and storage, or with the technology to process and analyze it.
Big data as a type of data: Transactions vs. Events

Transactions:
- Each one is valuable
- The elements of a transaction can be aggregated easily
- A set of transactions does not usually have important ordering or dependency
- Mutable

Events:
- A single event often has no value, e.g. what is the value of one click in a series? Some events are extremely valuable, but this is only detectable within the context of other events.
- Elements of events are often not easily aggregated
- A set of events usually has a natural order and dependencies
- **Immutable**
Example “big data”: Web tracking data

<table>
<thead>
<tr>
<th>USER_ID</th>
<th>301212631165031</th>
</tr>
</thead>
<tbody>
<tr>
<td>SESSION_ID</td>
<td>590387153892659</td>
</tr>
<tr>
<td>VISIT_DATE</td>
<td>1/10/2010 0:00</td>
</tr>
<tr>
<td>SESSION_START_DATE</td>
<td>1:41:44 AM</td>
</tr>
<tr>
<td>PAGE_VIEW_DATE</td>
<td>1/10/2010 9:59</td>
</tr>
<tr>
<td>DESTINATION_URL</td>
<td><a href="https://www.phisherking.com/gifts/store/logonForm?mmc=link-src-email-_m100109_-44IOJ1-_shop&amp;langId=-1&amp;storeId=1055&amp;URL=BECGiftListItemDisplay">https://www.phisherking.com/gifts/store/logonForm?mmc=link-src-email-_m100109_-44IOJ1-_shop&amp;langId=-1&amp;storeId=1055&amp;URL=BECGiftListItemDisplay</a></td>
</tr>
<tr>
<td>REFERRAL_NAME</td>
<td>Direct</td>
</tr>
<tr>
<td>REFERRAL_URL</td>
<td>-</td>
</tr>
<tr>
<td>PAGE_ID</td>
<td>PROD_24259_CARD</td>
</tr>
<tr>
<td>REL_PRODUCTS</td>
<td>PROD_24654_CARD, PROD_3648_FLOWERS</td>
</tr>
<tr>
<td>SITE_LOCATION_NAME</td>
<td>VALENTINE'S DAY MICROSITE</td>
</tr>
<tr>
<td>SITE_LOCATION_ID</td>
<td>SHOP-BY HOLIDAY VALENTINES DAY</td>
</tr>
<tr>
<td>IP_ADDRESS</td>
<td>67.189.110.179</td>
</tr>
<tr>
<td>BROWSER_OS_NAME</td>
<td>MOZILLA/4.0 (COMPATIBLE; MSIE 7.0; AOL 9.0; WINDOWS NT 5.1; TRIDENT/4.0; GTB6; .NET CLR 1.1.4322)</td>
</tr>
</tbody>
</table>

“unstructured” data embedded in the logged message: complex strings
The missing ingredient from most big data

METADATA!
More data: patterns emerge from lots of event data

Patterns emerge from the underlying structure of the *entire dataset*. The patterns are more interesting than sums and counts of the events.

Web paths: clicks in a session as network node traversal.

Email: traffic analysis producing a network

The event stream is a source for analysis, *generating another set of data* that is the source for different analysis.
Unstructured data isn’t really unstructured: events have structure, language has structure. Text can contain traditional structured data elements. The problem is that the content is unmodeled.
Big changes for data warehousing workloads

The results of analytic processing can, often do, feed back into the system from which they originate.

Much of the data is being read, written and processed in real time.

Our design point was not real time ingest, changing tables and ephemeral patterns.
Focus on one thing: workloads

The single most important aspect of technology suitability
There are really **three** workloads to consider, not two

1. **Operational**: OLTP systems
2. **Analytic**: OLAP systems
3. **Algorithmic**: Processing systems

Unit of focus:

1. Transaction
2. Query
3. Algorithm

Different problems require different platforms
<table>
<thead>
<tr>
<th>Workloads</th>
<th>OLTP</th>
<th>BI</th>
<th>Analytics</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Access</strong></td>
<td>Read-Write</td>
<td>Read-only</td>
<td>Read-mostly</td>
</tr>
<tr>
<td><strong>Predictability</strong></td>
<td>Predictable</td>
<td>Unpredictable</td>
<td>Fixed path</td>
</tr>
<tr>
<td><strong>Selectivity</strong></td>
<td>High</td>
<td>Low</td>
<td>Low</td>
</tr>
<tr>
<td><strong>Retrieval</strong></td>
<td>Low</td>
<td>Low</td>
<td>High</td>
</tr>
<tr>
<td><strong>Latency</strong></td>
<td>Milliseconds</td>
<td>&lt; seconds</td>
<td>msecs to days</td>
</tr>
<tr>
<td><strong>Concurrency</strong></td>
<td>Huge</td>
<td>Moderate</td>
<td>1 to huge</td>
</tr>
<tr>
<td><strong>Model</strong></td>
<td>3NF, nested object</td>
<td>Dim, denorm</td>
<td>BWT</td>
</tr>
<tr>
<td><strong>Task size</strong></td>
<td>Small</td>
<td>Large</td>
<td>Small to huge</td>
</tr>
</tbody>
</table>
An (overly) Simple Division of the Problem Space

- **Big analytics, little data**: Specialized computing, modeling problems: supercomputing, GPUs
- **Little analytics, little data**: The entry point; SAS, SMP databases, even OLAP cubes can work
- **Big analytics, big data**: Complex math over large data volumes requires distributed data engines
- **Little analytics, big data**: The BI/DW space, for the most part, with work done in databases

© Third Nature Inc.
TANSTAAFL

When replacing the old with the new (or ignoring the new over the old) you always make tradeoffs, and usually you won’t see them for a long time.

Technologies are not perfect replacements for one another. Often not better, only different.
Which is best?, 3NF or dimensional?

The core assumption that there can be just one big schema model on one big platform is flawed. Workloads are different.

Answer: neither.

We think we can model all the data before use, but that’s a bottleneck. Current techniques for modeling and managing data are too rigid and incapable of describing all the possible relationships.
We have a design for stability. We need one for adaptability.
A core problem with a single schema is change
The big data answer: schema on read

Match the shape to the hole

or

Match the hole to the shape

Predicate schemas for write flexibility (agility) and speed
There's a price to pay with using "schema-on-read" for everything. You won't see the problems with this until you add a second application, and a third. "One writer-many readers" kills schema-on-read benefits.

Not flexibility vs control, but vs repeatability
When to use implicit schema?

Use **implicit** (on read) when:

▪ You can hide the persistence of your data behind a service
▪ Nobody will ever want access to that data except you
▪ When data dies with the code
▪ You need to write data at a very high rate
▪ Your data sources change or are variable

Use **explicit** (on write) when:

▪ you need to send data to another application
▪ when more than one application (or person) needs to use data
▪ when data lives longer than your code
▪ When the data is regular
▪ When the sources and structure do not change
▪ When querying is more important than writing
The market is cyclical: databases in No-tation

1970: NoSQL = We have no SQL
1980: NoSQL = Know SQL
2000: NoSQL = No SQL?
2005: NoSQL = No SQL!
2010: NoSQL = Not only SQL
2015: NoSQL = No, SQL!

(R)DB(MS)
Maybe...

These aren’t the databases we’re looking for.

We need one that speaks pig.
One way Hadoop is a lot like databases: all alike, yet each is a unique special snowflake.
Hadoop: a summary of the magic

1. Provides both storage and complex processing as part of the same platform
2. Makes parallel programming more accessible
3. Schemaless (just files) therefore flexible
4. Inexpensive, reliable scale-out
5. Potential for fast, scalable data ingest

The bad stuff:
- Concurrency
- Not great for mutable data
- Mostly file-based sequential processing, or you store data many times in different datastores (locality is important)
- Minimal data management (today)
Reality: Hadoop disaggregates the database

One of the key things Hadoop does is separate the storage, execution and API layers of a database. This allows for processing flexibility, but it does not permit one to build a reliable, high performance database across the layers.

Abstraction layers

General-purpose data engines

Storage management

Hadoop distributed filesystem (HDFS)
A more specific look at layers and engines

You can program to any layer you choose. Some projects already build on top of multiple others.

- **Base storage**
  - Hadoop distributed filesystem (HDFS)
  - Storage (filetypes in HDFS, Hbase, etc)

- **Storage mgmt**
  - Hadoop distributed filesystem (HDFS)

- **Engine**
  - MapReduce
  - Spark
  - Tez
  - Cascading
  - Kylin
  - Phoenix

- **Abstraction layer / API**
  - Native API
  - Crunch
  - Pig
  - Hive
  - Giraph
  - Native API
  - Crunch
  - Pig
  - Hive
  - Native API
  - SparkSQL
  - Native API
  - Pig
  - Hive
  - Native API
  - Impala
  - Drill
  - Presto
  - Hbase

**Language/API**
- SQL, MDX
- Language/API
- Engine
An important Hadoop + cloud computing benefit

**Scalability is free** – if your task requires 10 units of work, you can decide when you want results:

- 1 server, 10 units of time
- 10 servers, 1 unit of time

*Cost is the same. Not true of the conventional IT model.*
Four core capabilities big data technologies add

1. Unlimited scale of storage, processing
   - Agility, faster turnaround for new data requests (but not a replacement for BI)
   - Fewer staff to accomplish same goals

2. New data accessibility
   - More data retained for longer period
   - Access to data unused due to cost or processing limits
   - Any digital information becomes usable data

3. Scalable realtime processing
   - Brings ability to monitor and act on data as events occur

4. Arbitrary processing, analytics
   - Faster analysis
   - Deeper analysis
   - More broadly accessible analytics
The solution to our problems isn’t technology, it’s architecture.
The geography has been redefined

The box we created:

• not any data, **rigidly typed data**
• not any form, **tabular rows and columns of typed data**
• not any latency, **persist what the DB can keep up with**
• not any process, **only queries**

The digital world was diminished to only what’s inside the box until we forgot the box was there.
In the DW world both data and processing are bounded:

- No consideration for feedback loops and change.
- Processing only happens here.
- Nobody here creates new information.
- Sources few and well understood.
- Complex DI is controlled by IT.
- Schemas are few and designed.
- Tools are authorized, few in number and kind.

This is a monolithic, layered architecture.
Break down the monolithic architecture
Deconstructing data environments

There are three things happening in a data warehouse:

- Data acquisition
- Data management
- Data delivery

Isolate them from one another, allow read-write use, and you are on the path.
Data lake / LDW / AE components

In reality, you are building three systems, not one. Avoid the monolith.
Data lake functions depend on platform services

**Data Acquisition**
- Collect & Store

**Data Management**
- Process & Integrate

**Data Access**
- Deliver & Use

**Workflow Management**

**Processing Engines**

**Data Movement**

**Data Curation**

**Data Persistence**

**Dataflow Services**

**Metadata**

**Base Platform Services**

© Third Nature Inc.
We’re so focused on the light switch that we’re not talking about the light

**DATA ARCHITECTURE**
Always design for change Isolation
Decouple the Data Architecture

The core of the data warehouse isn’t the database, it’s the data architecture that the database and tools implement.

We need a data architecture that is not limiting:

- Deals with change more easily and at scale
- Does not enforce requirements and models up front
- Does not limit the format or structure of data
- Assumes the range of data latencies in and out, from streaming to one-time bulk
Food supply chain: an analogy for data

Multiple contexts of use, differing quality levels

You need to keep the original because just like baking, you can’t unmake dough once it’s mixed.
Data architecture is required by the services, and vice versa.

Data Management

- Process & Integrate

Data Acquisition

- Collect & Store

Data Access

- Deliver & Use

Platform Services

- Raw data in an immutable storage area
- Transient data
- Standardized or enhanced data
- Common or usage-specific data

© Third Nature Inc.
The data areas map (mostly) to functional areas

Collection can’t be limited by database scale and latency. Immutability, persistence and concurrency are required.

Collect  Manage & Integrate  Process, Deliver, Use

Real time

Incremental

Batch

One-time copy
Proper architecture enables evolutionary design for data

Evolutionary design is required because data needs change. You need a system not for stability – we have that in the DW - but for evolution and change.

Data Acquisition
- Collect & Store
  - Real time
  - Incremental
  - Batch
  - One-time copy

Data Management
- Process & Integrate

Data Access
- Deliver & Use

Data Lake Platform Services

You can’t build this all at once. You need to grow it over time.
BI is a commodity, a cost of doing business
Think like an architect, not like a consumer

No more “enterprise standard” – now it’s all about “what works”

The technology providers are selling you what they have, not what you need.

Follow the goals of the business.

Translate the goals into capabilities and match those to the architecture required.
How we develop best practices: survival bias

We don’t need best practices, we need worst failures.

© Third Nature Inc.
Conclusions

• Big data is an opportunity to modernize, take advantage of it.
• You do need new tech, don’t delude yourself.
• You still need most of the old tech, it works.
• Architecture is key: deconstruct what’s wrong, define the new, build toward it selectively.
• Changing methods will change architecture, you can’t build the new using the old methods.
• Agility and continuous delivery go together.
• People are more important than products.
“When a new technology rolls over you, you're either part of the steamroller or part of the road.” – Stewart Brand
Thanks to the people who supplied the images used in this presentation:

- Indonesian angry mask phone: [Erik De Castro](http://www.flickr.com/photos/thomashawk/2670706781/)
- Round hole square peg: [https://www.flickr.com/photos/epublicist/3546059144](https://www.flickr.com/photos/epublicist/3546059144)
- Building demolition: [https://www.flickr.com/photos/gregpc/4429888820](https://www.flickr.com/photos/gregpc/4429888820)

© Third Nature Inc.
Mark Madsen is president of Third Nature, a consulting and advisory firm focused on analytics, business intelligence and data management. Mark is an award-winning author, architect and CTO. He is an international speaker, a contributor to numerous publications, and member of the O’Reilly Strata program committee. For more information or to contact Mark, follow @markmadsen on Twitter or visit http://ThirdNature.net
About Third Nature

Third Nature is a consulting and advisory firm focused on new and emerging technology and practices in information strategy, analytics, business intelligence and data management. If your question is related to data, analytics, information strategy and technology infrastructure then you’re at the right place.

Our goal is to help organizations solve problems using data. We offer education, consulting and research services to support business and IT organizations as well as technology vendors.

We fill the gap between what the industry analyst firms cover and what IT needs. We specialize in strategy and architecture, so we look at how technologies are applied to solve problems rather than evaluating product features.