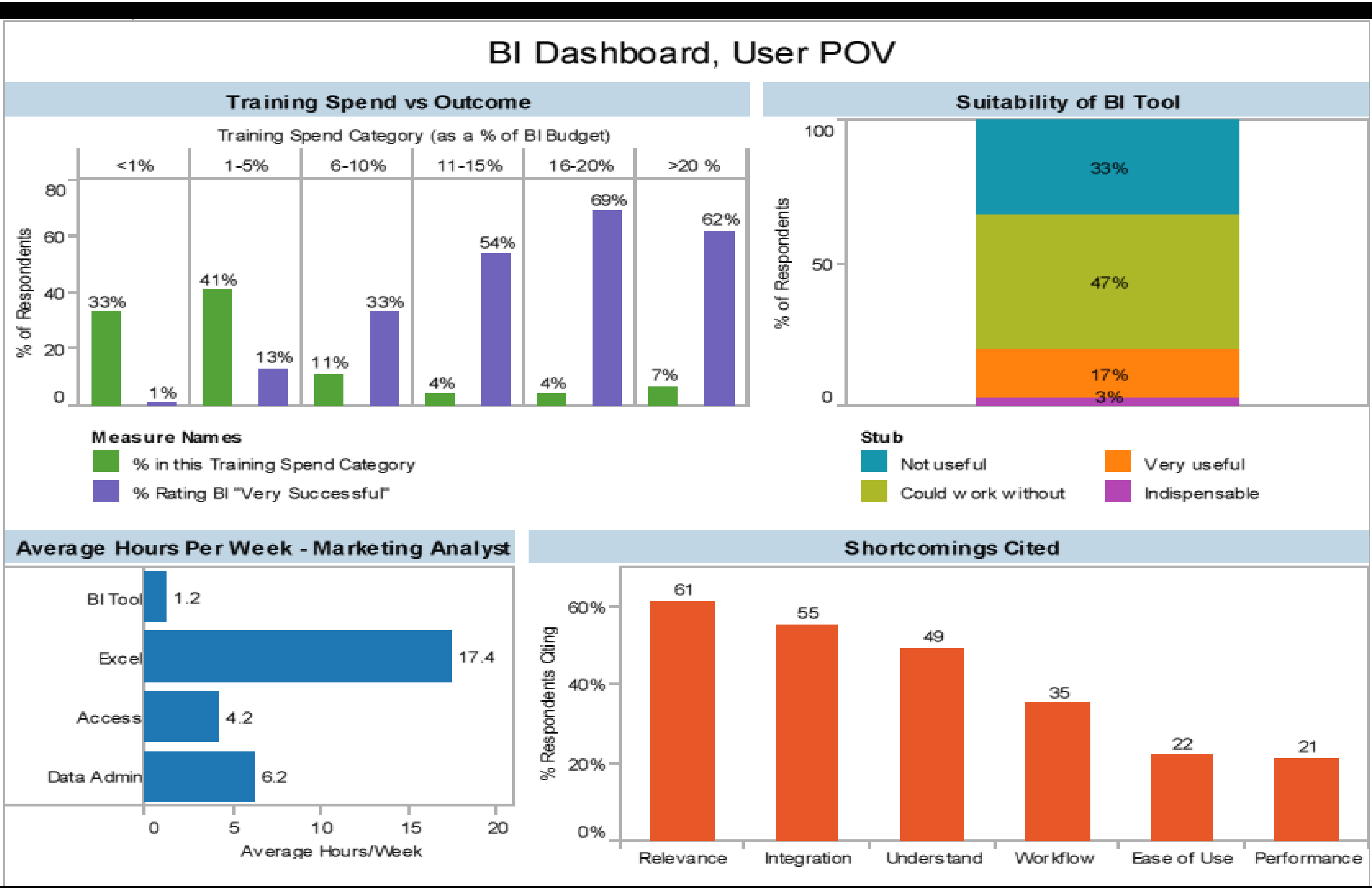
BI on Big Data

Strata San Jose, March, 2018

Shant Hovsepian, *Arcadia Data*

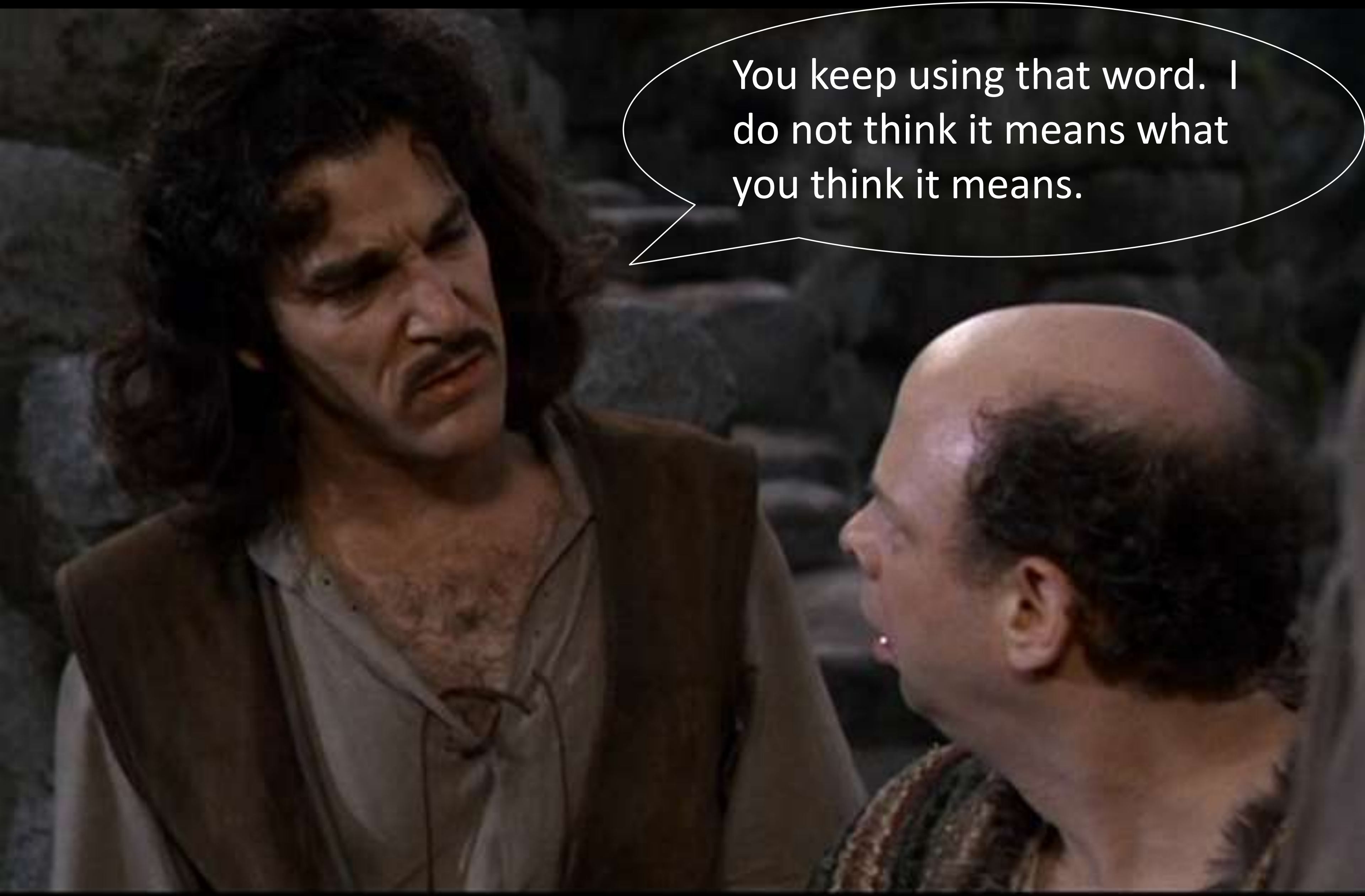
Mark Madsen, *Think Big Analytics*

Users hate BI. And have for a long time



The old problem was access, the new problem is analysis

What do you mean by “analytics”?


You keep using that word. I do not think it means what you think it means.

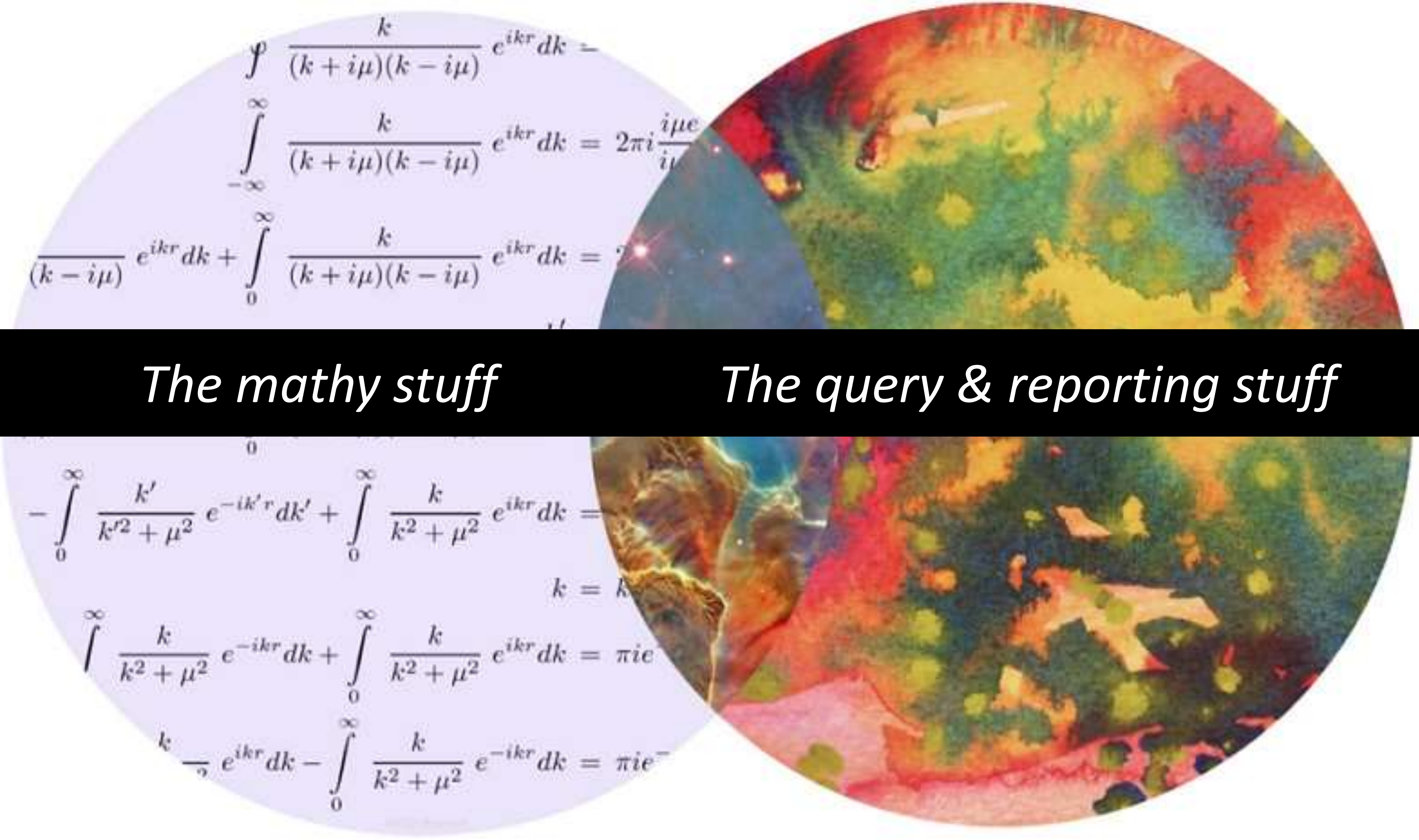
User-focused criteria: context and point of use

Information use is diverse and varies based on context:

- Get a quick answer
- Solve a one-off problem
- Analyze causes
- Do experiments
- Make repetitive decisions
- Use data in routine processes
- Make complex decisions
- Choose a course of action
- Convince others to take action

One size doesn’t fit all.

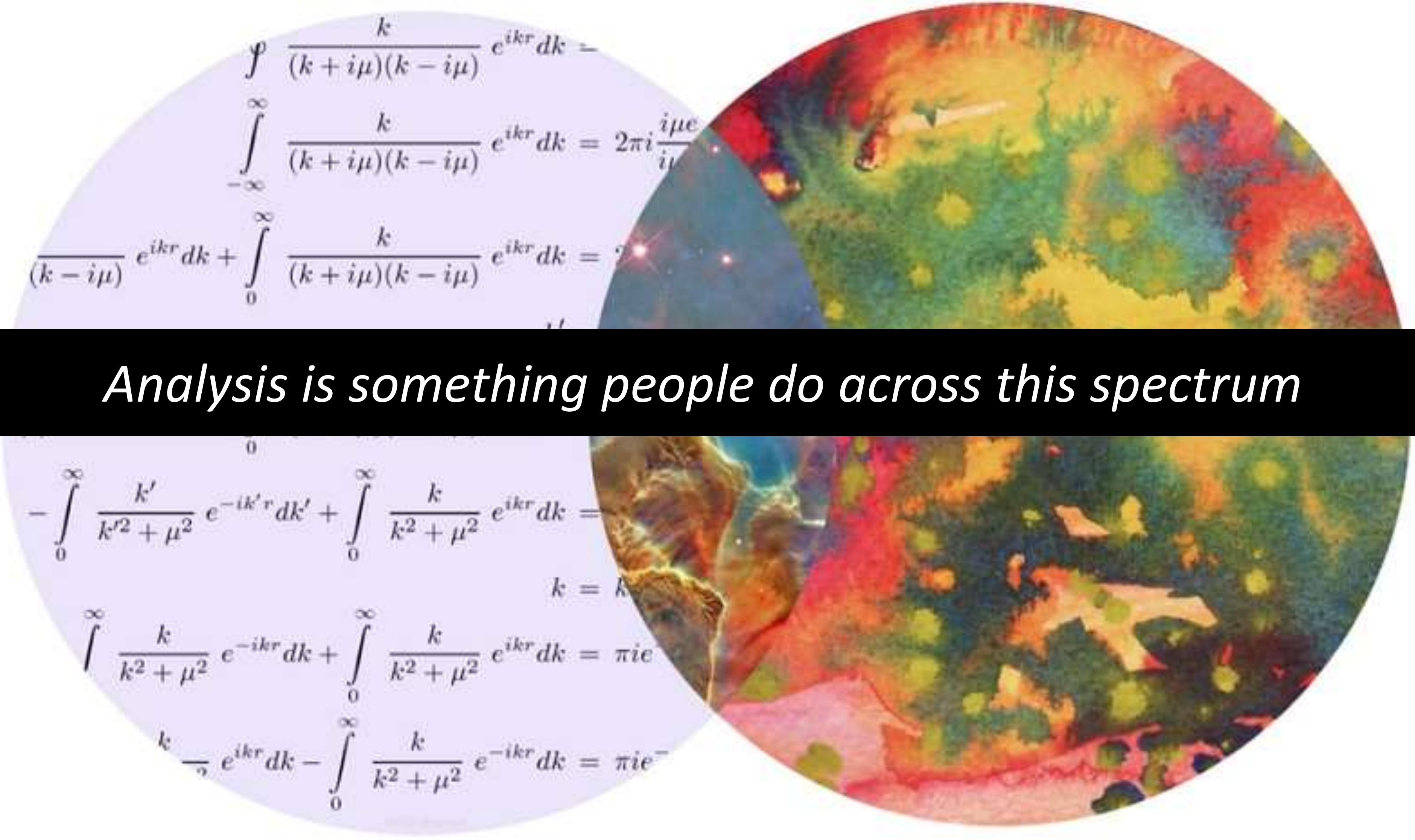
There are two parts to “analytics”



The mathy stuff

The query & reporting stuff

Analysis: the verb that’s ignored



Analysis is something people do across this spectrum

Good paper on the topic of analysis: Enterprise Data Analysis and Visualization: An Interview Study
http://vis.stanford.edu/files/2012-EnterpriseAnalysisInterviews-VAST.pdf

Old market says: There's nothing wrong with what you have, just keep buying new products from us

The big data market has an answer…

The data lake: just dump the data in!

Combine with self-service tools: we'll figure it out later!

GOOD NEWS EVERYONE
WE'RE BACK WHERE WE STARTED!

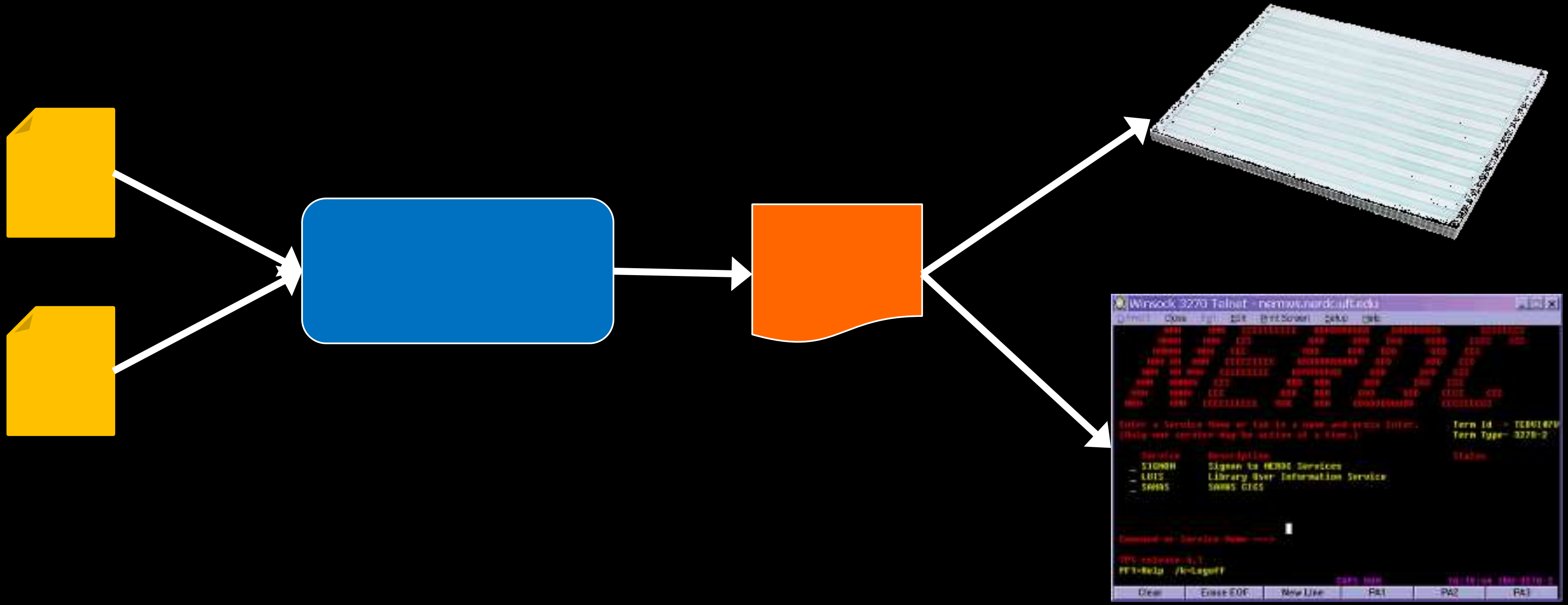
The primary view of BI, self service is publishing data

An architectural history of BI tools

First there were files and reporting programs.

Application files feed through a data processing pipeline generate an output file. The file is used by a report formatter for print/screen.

Every report is a program written by a developer.

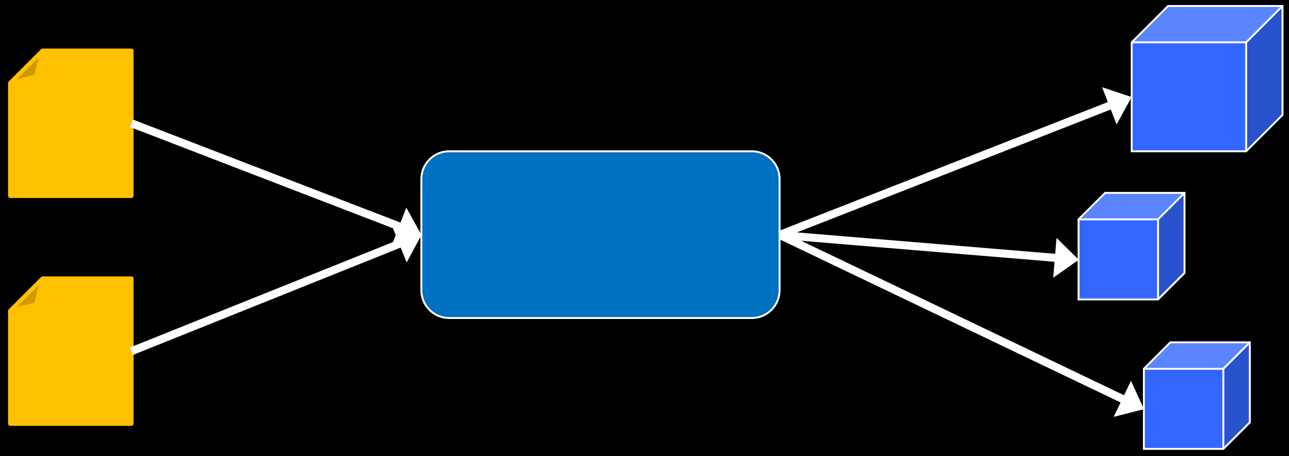


```
                                                  Term Id  - TE[illegible]
                                                  Term Type- 3278-2

   _ SIGNON      Signon to NERDC Services
   _ LUIS        Library User Information Service
   _ SAMAS       SAMAS CICS

PF1=Help  /k=Logoff
```

An architectural history of BI tools

We had the concept of cubes before we had RDBMSs. First commercial product (Express) in 1970.

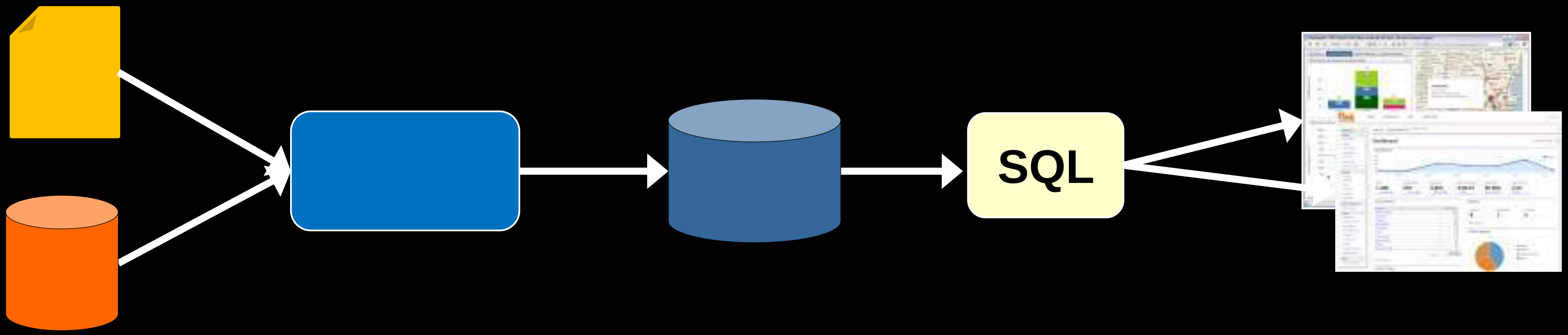
Provided interactive response time but had problems: rigidly defined schema, inflexible definition, data size limits, slow cube build times, resulting in cube explosions.



An architectural history of BI tools

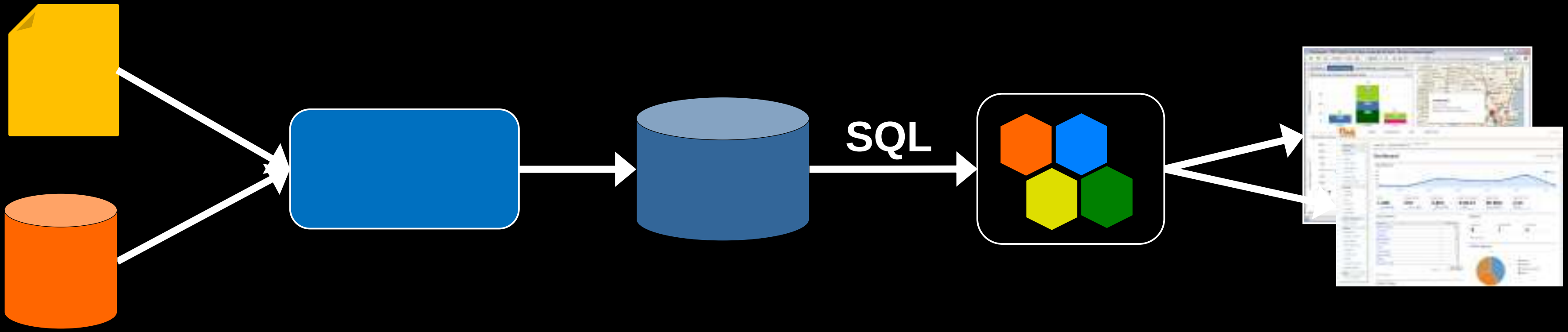
Then we had databases and tools with embedded SQL, and query-by-example templating soon after.

These had better scalability and flexibility over OLAP models. They decoupled storage from access from the rendering of the interface.

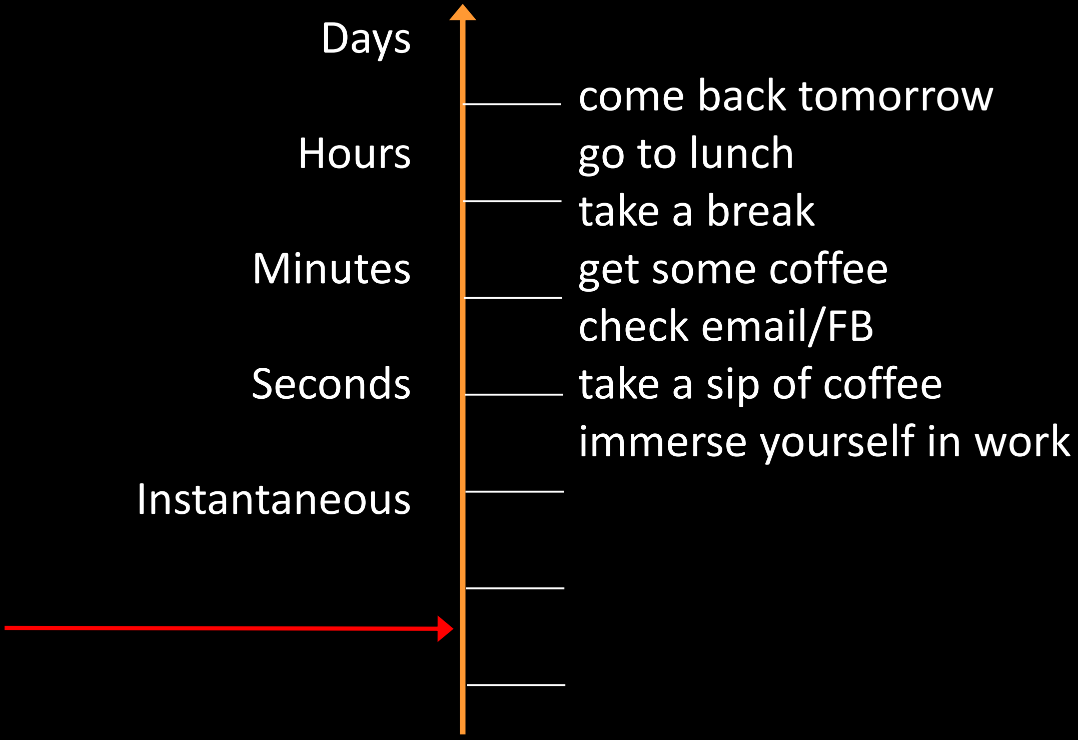


An architectural history of BI tools

With more regular schema models, in particular dimensional models that didn’t contain cyclic join paths, it was possible to automate SQL generation via semantic mapping layers. Query via business terms made BI usable by non-technical people.



Response time is a key driver for analysis



Days

Hours

Minutes

Seconds

Instantaneous

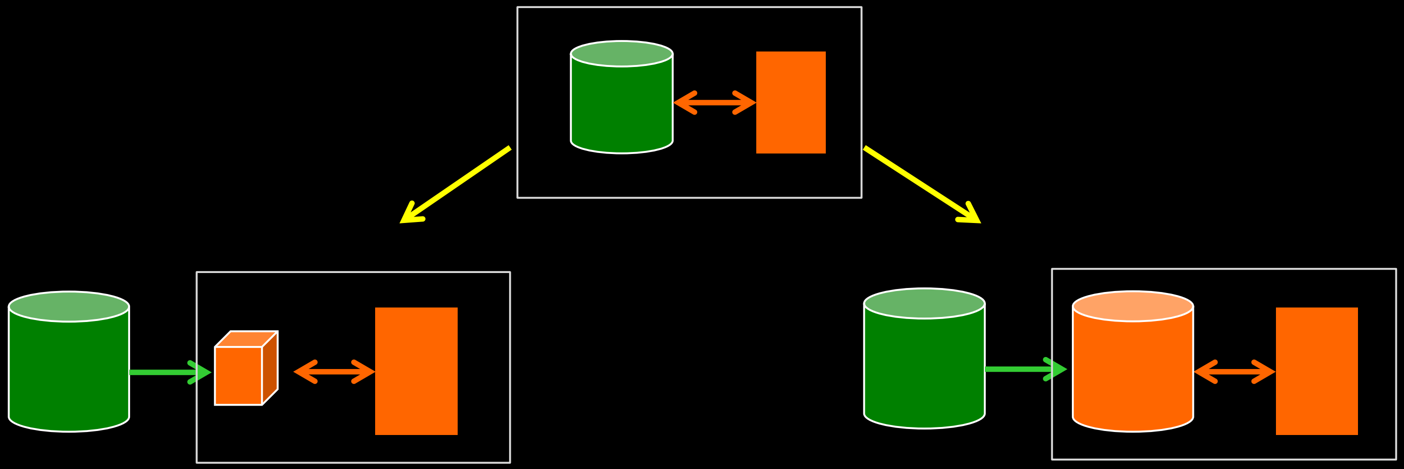
come back tomorrow
go to lunch
take a break
get some coffee
check email/FB
take a sip of coffee
immerse yourself in work

Flow is possible only in the “less than 3 second” range

Many approaches today are a step backward. Unless you resolve this task performance gap, real analysis work is a challenge and will remain the domain of Excel and tools that extract subsets of data into memory.

BI server architecture shifts

The SQL-generating server model of BI scales extremely well but has poor user response time.

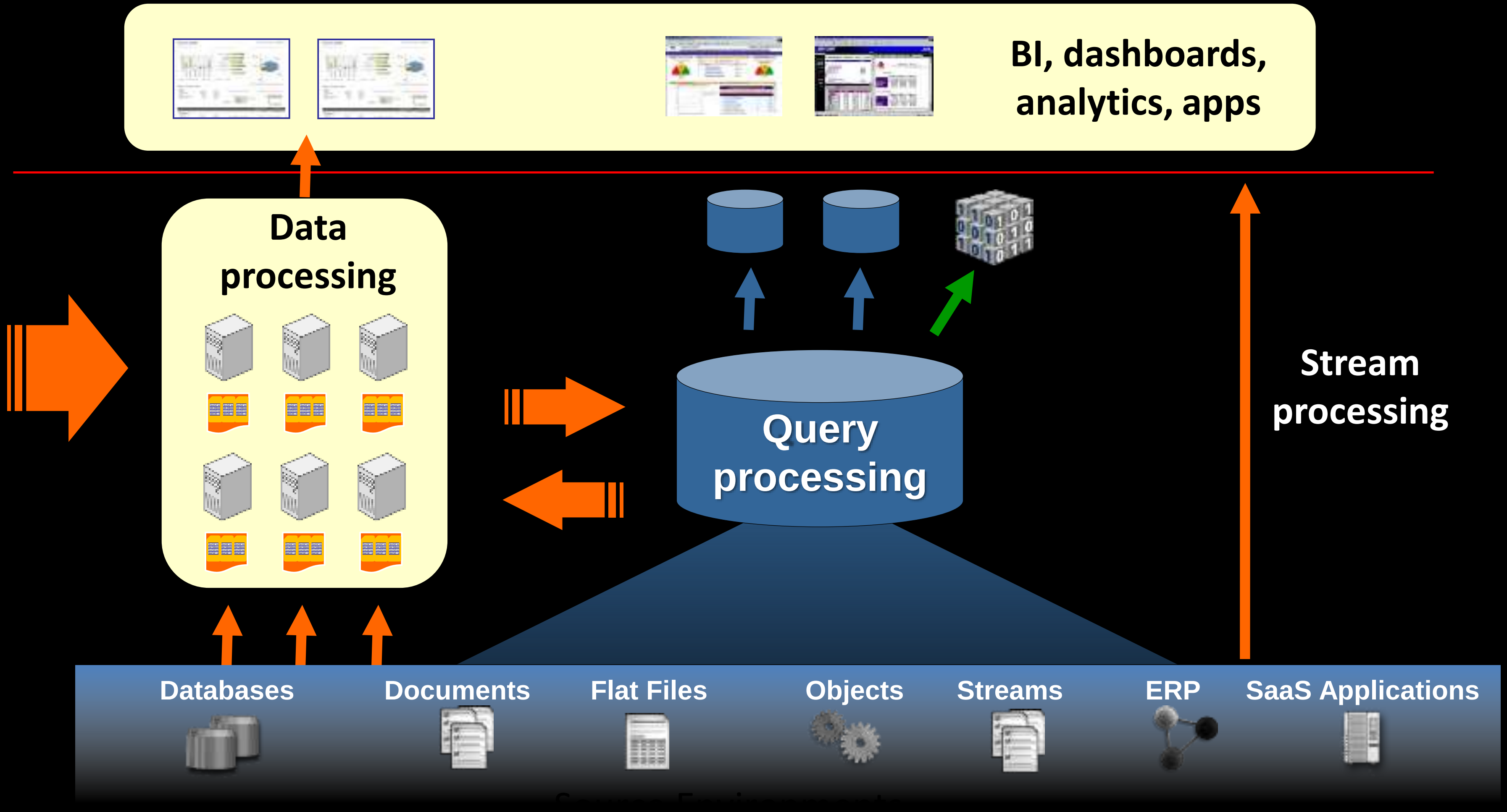


Solution 1: pre-cache query results or prebuild datasets on the BI server (i.e. the old OLAP model) Well-known problems with this.

Solution 2: Shove all the data into a BI server repository. Avoids subset problems. Adds potential scaling problems.

IT reality today is multiple data stores, not one place

Independent, purpose-built databases and processing systems for different types of data and query / computing workloads is the new norm for information delivery. No single place for data or access.



There is always a third way

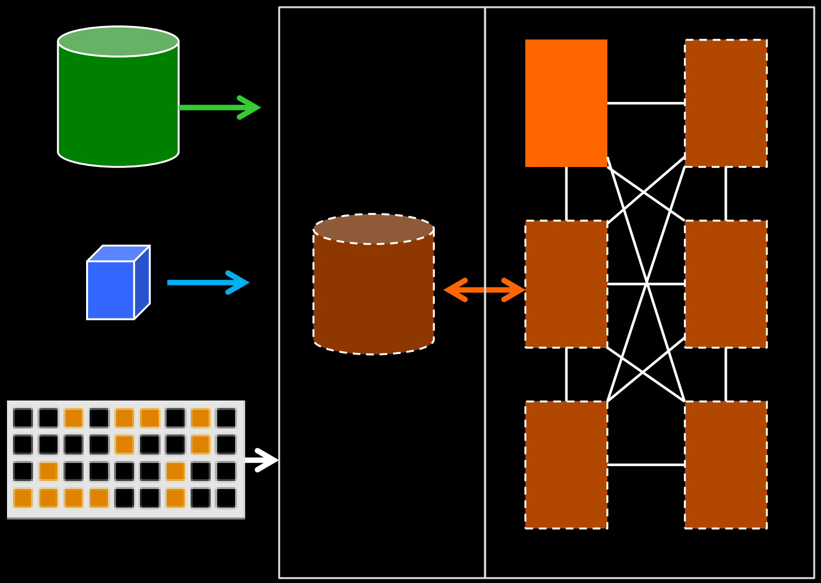
The previous choices were driven by client-server thinking. We have a distributed (cloud) environment.

Possibilities:

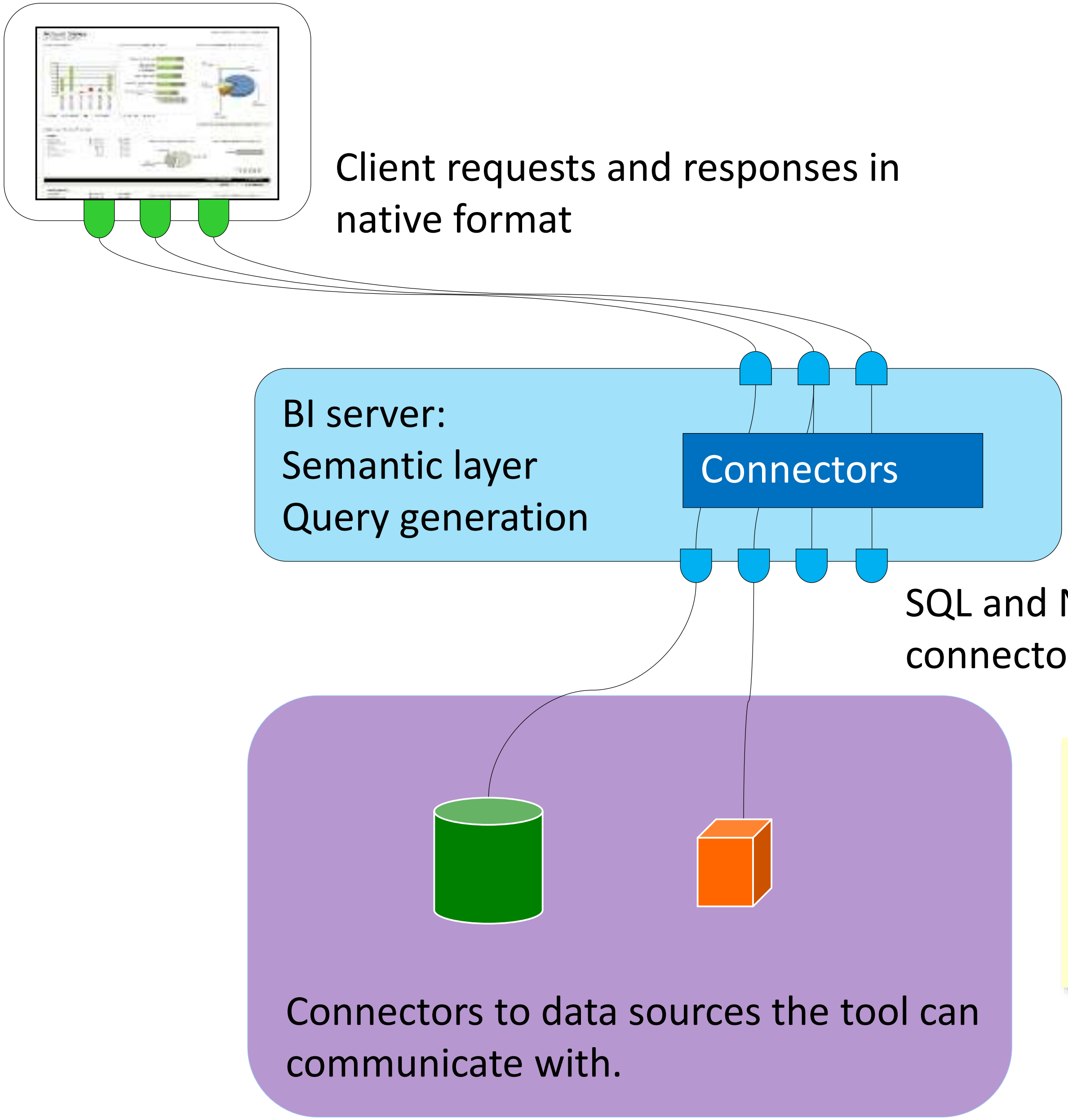
Don’t force all the compute into the DB or server.

Don’t force all the compute to the client.

Data on demand, bring it to the analysis from where it is, and/or execute the analysis local to where the data is.



Semantic layer tools: what we're used to



Client requests and responses in native format

BI server:
Semantic layer
Query generation

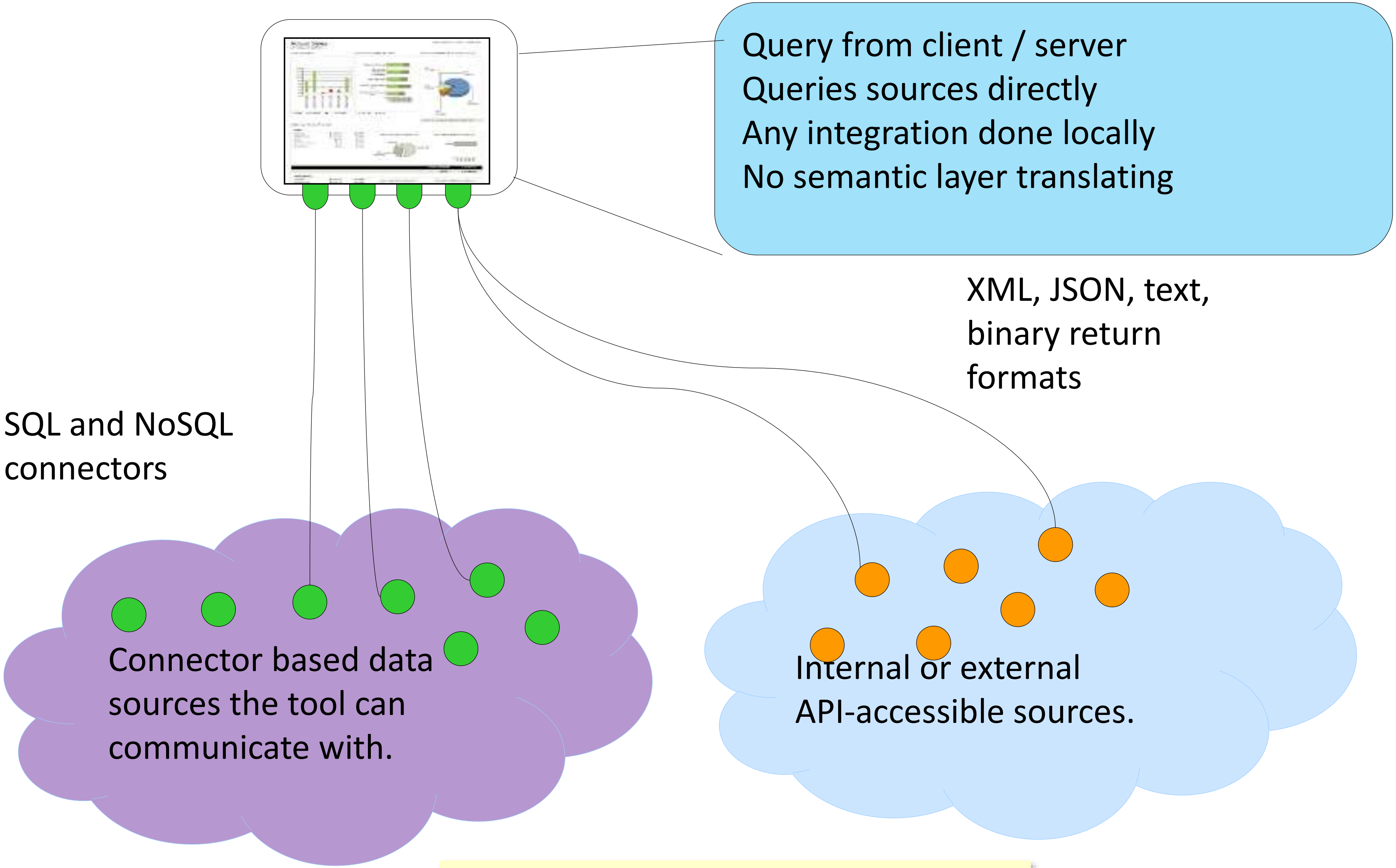
Connectors

SQL and NoSQL connectors

Connectors to data sources the tool can communicate with.

Note: most BI servers still can't talk to more than one DB at the same time

Map-based tools: what we’re back to



Query from client / server
Queries sources directly
Any integration done locally
No semantic layer translating

XML, JSON, text, binary return formats

SQL and NoSQL connectors

Connector based data sources the tool can communicate with.

Internal or external API-accessible sources.

Note: most products have minimal to no local join ability

With analysis, the BI approach has to be inverted

The process used today for data warehousing:

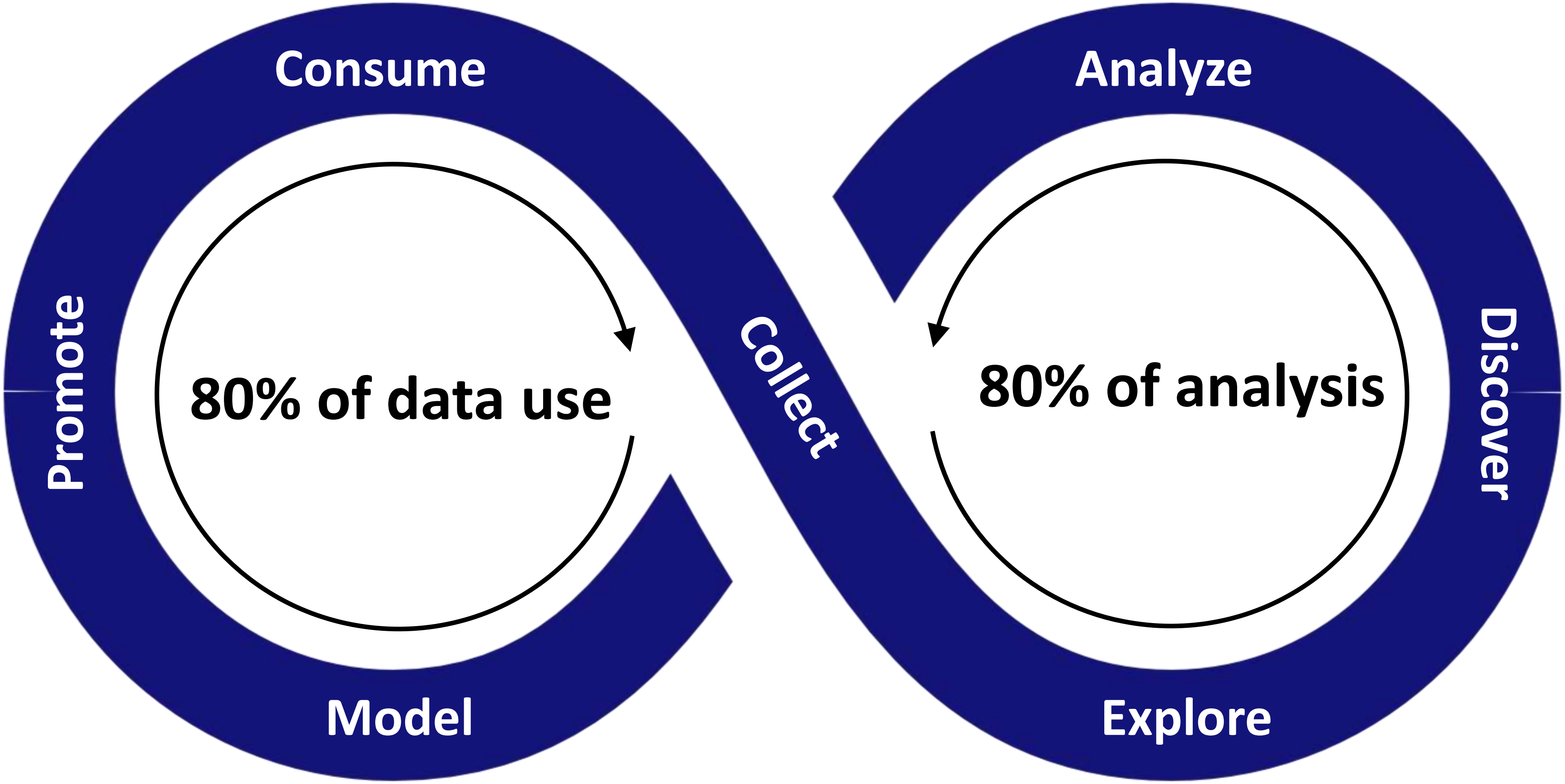
1. Model
2. Collect
3. Analyze

The new process is:

1. Collect
2. Analyze
3. Model

This is a shift from *planned design* to *adaptive design* for data management, and a multi-skill team.

Analysis and BI: Discoverability and Repeatability



Focus on repeatability

Application cycle time

Focus on discoverability

Analyst cycle time

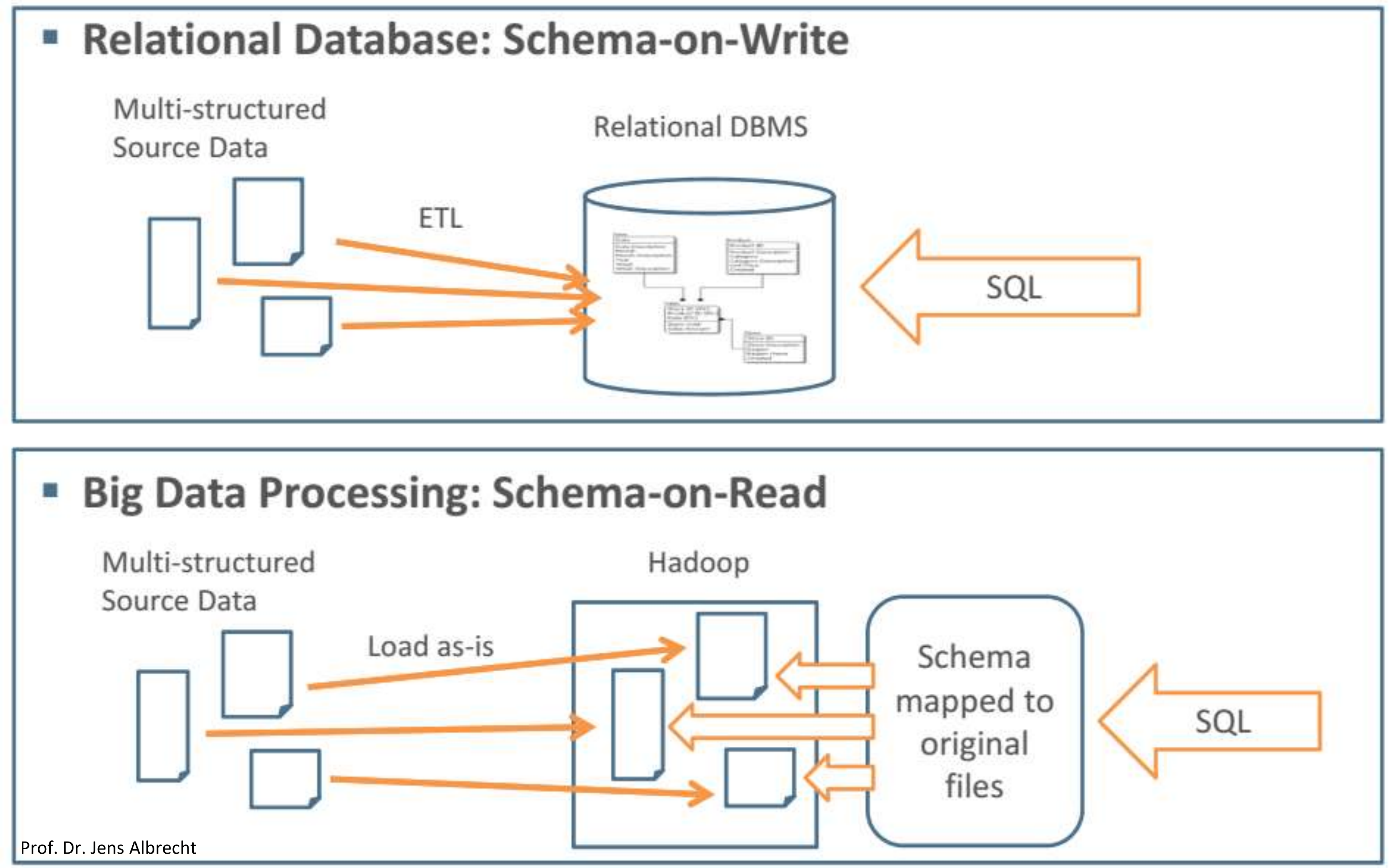
Just enough modeling: an analogy for analytic data

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.

A core problem with one global schema is change

Big data answer? Schema on read



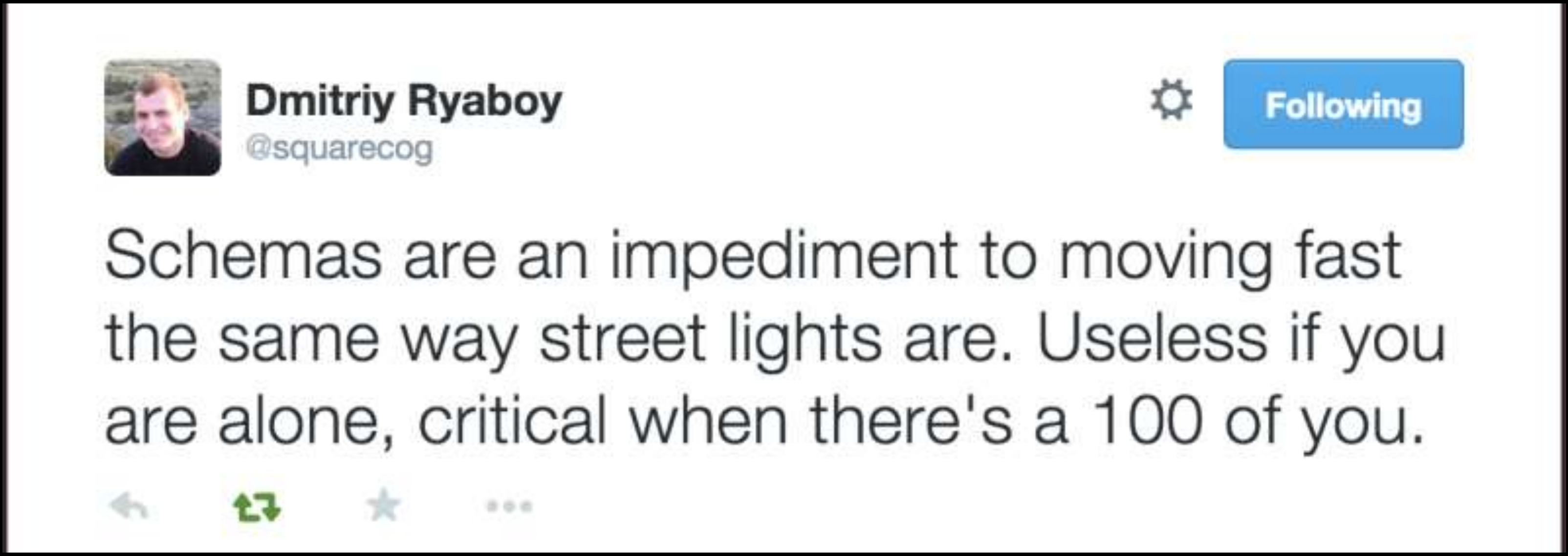
- Relational Database: Schema-on-Write

- Big Data Processing: Schema-on-Read

Prof. Dr. Jens Albrecht

Schema on read is an answer, sometimes

Schema-on-read is really only good for the developer who doesn’t know what to do with the data. There is a price to pay with schema on read, but you usually don’t see it at the beginning.



Schemas are an impediment to moving fast the same way street lights are. Useless if you are alone, critical when there's a 100 of you.

SoR problem: metadata is what you wished your data looked like.

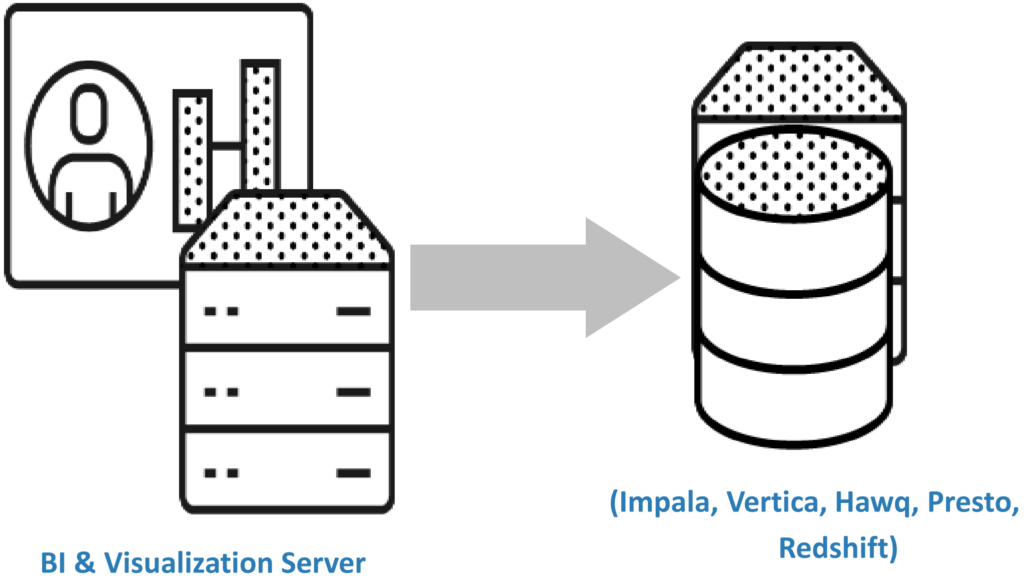
Reality is *not* requirements = code

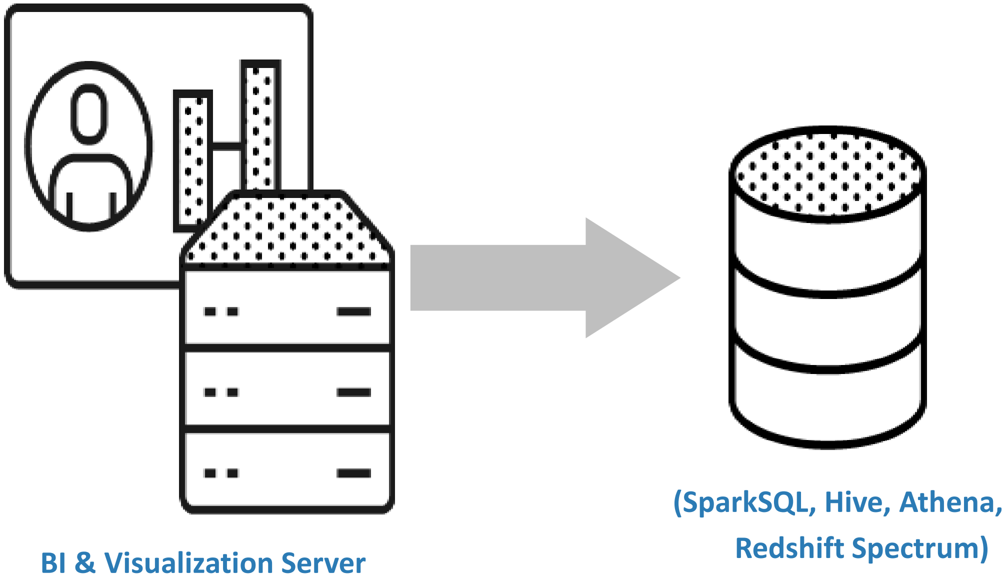
Reality is the data, not the metadata

How did we get to this point with BI & big data?

There’s a difference between having no past and actively rejecting it.

ODBC to SQL in Big Data, SQL on Big Data


(Impala, Vertica, Hawq, Presto, Redshift)
BI & Visualization Server


(SparkSQL, Hive, Athena, Redshift Spectrum)
BI & Visualization Server

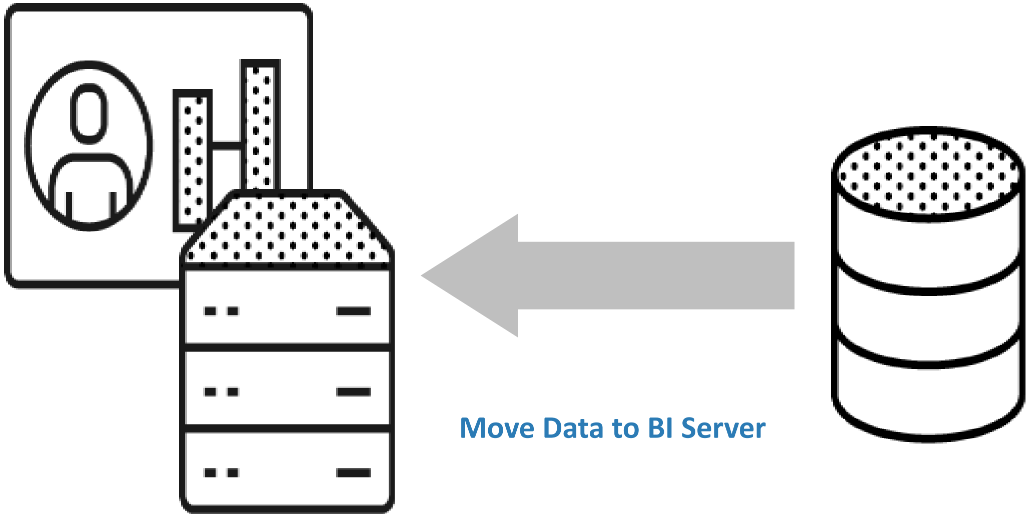
Pros

- ✓ Can get detailed data
- ✓ Performance leverages the architecture
- ✓ You get all of SQL

Cons

- ✗ Lower user concurrency
- ✗ Cannot access unstructured data (requires schema)
- ✗ Cost - Manage security in multiple tools, separate administration for metadata

Move Data to Separate BI Server



Move Data to BI Server

BI & Visualization Server

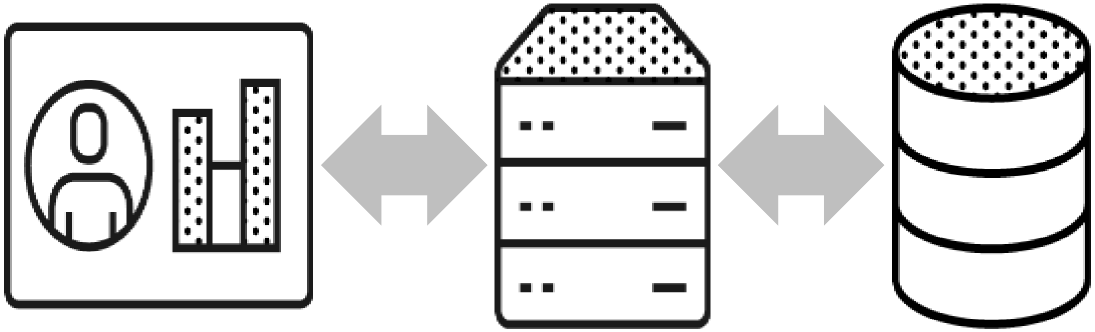
Pros

- ✓ Least Costly
- ✓ Use existing BI tools
- ✓ Full functionality without compromise

Cons

- ✗ Shallow insights – summary data
- ✗ Scale single server
- ✗ IT required Data movement
- ✗ Separate security models
- ✗ Batch data updates

(R)OLAP on Big Data



OLAP Middleware

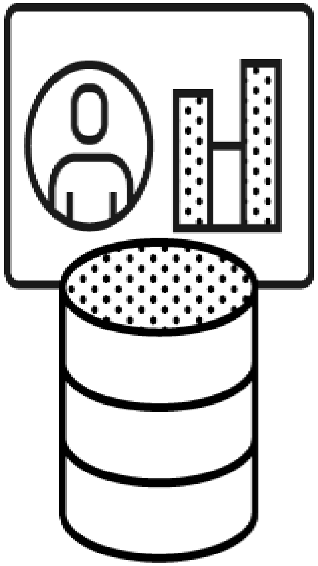
Pros

- ✓ Use existing BI tools
- ✓ Higher user concurrency
- ✓ Predictable response time

Cons

- ✗ Lacks ad-hoc freedom
- ✗ Not real-time: batch data updates
- ✗ Very rigid schema
- ✗ Multiple tools and data duplication
- ✗ Separate security models, administration

Native BI



BI runs on Big Data

Pros

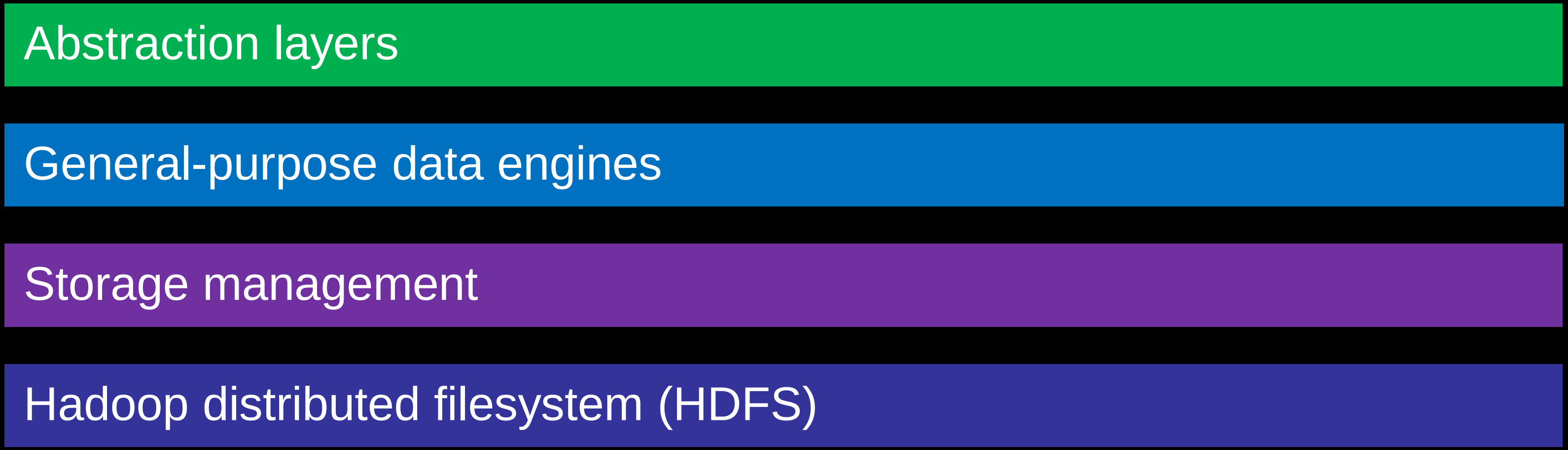
- ✓ More user concurrency
- ✓ Linear scalability
- ✓ Agility for analysts (drill to detail)
- ✓ Supports direct complex data sources

Cons

- ✗ Newer technology and approach
- ✗ Requires some Big Data skills to set up and maintain
- ✗ Workload management needed to response time

Hadoop: it disaggregates the database

One of the key things Hadoop does is to 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. You trade these for write flexibility.



Abstraction layers

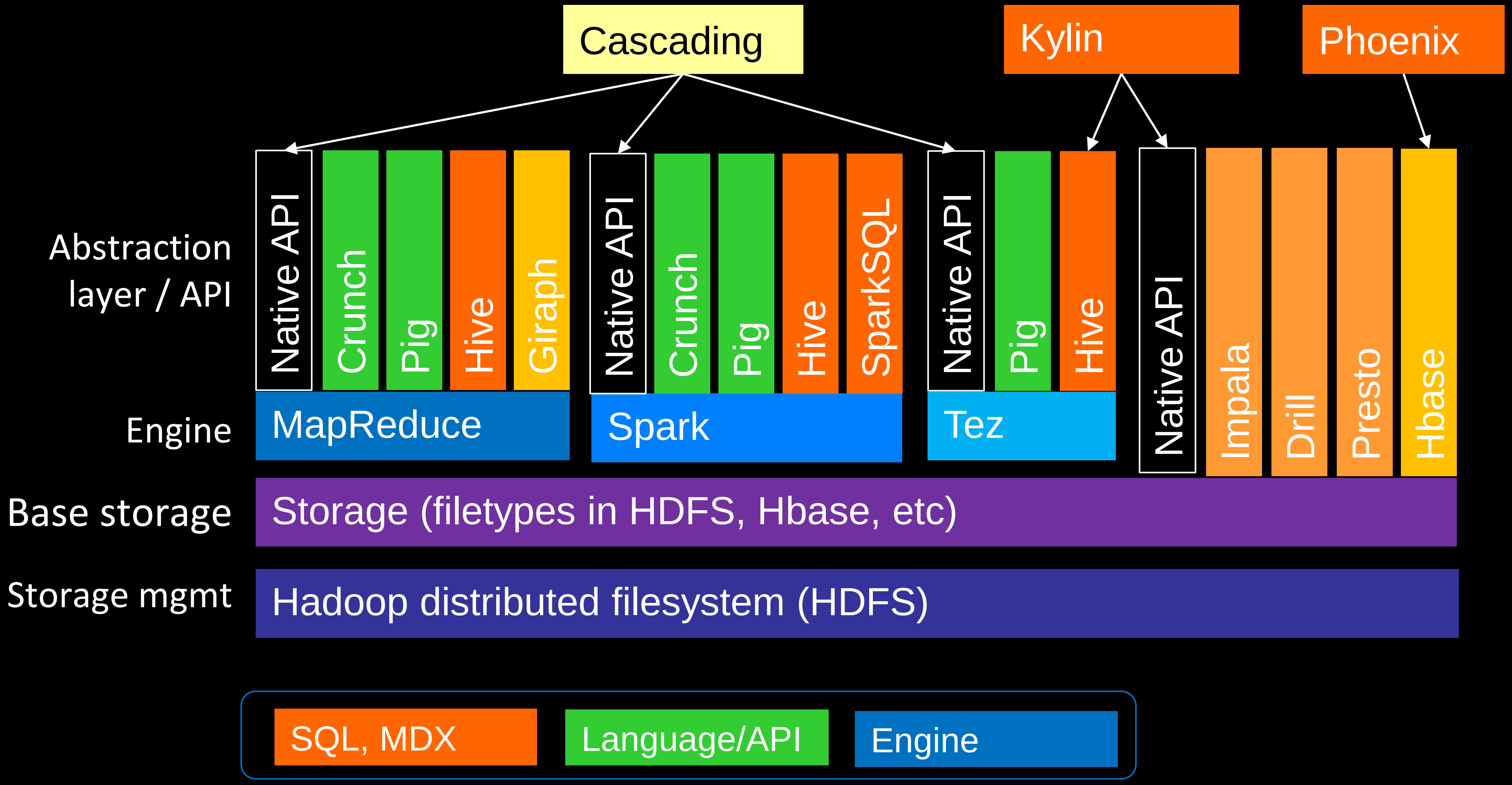
General-purpose data engines

Storage management

Hadoop distributed filesystem (HDFS)

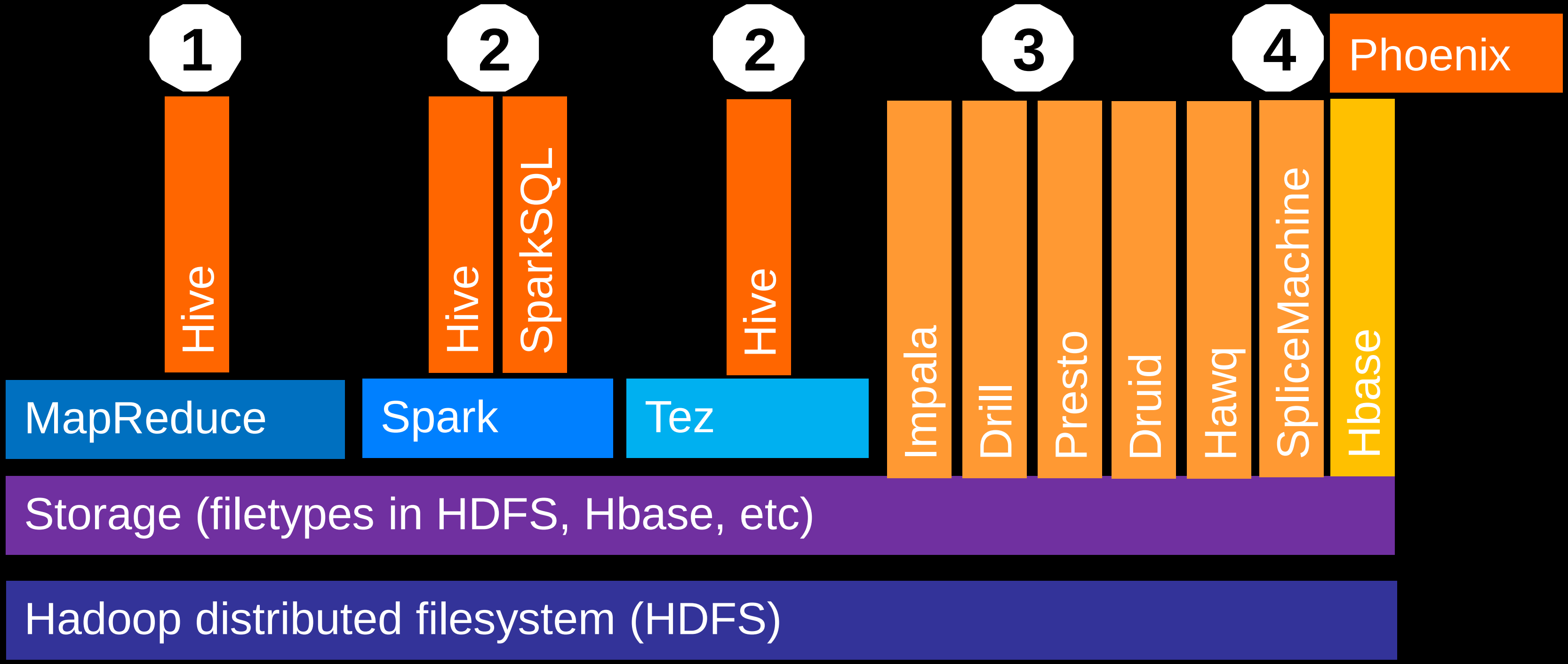
A more specific look at layers and engines

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

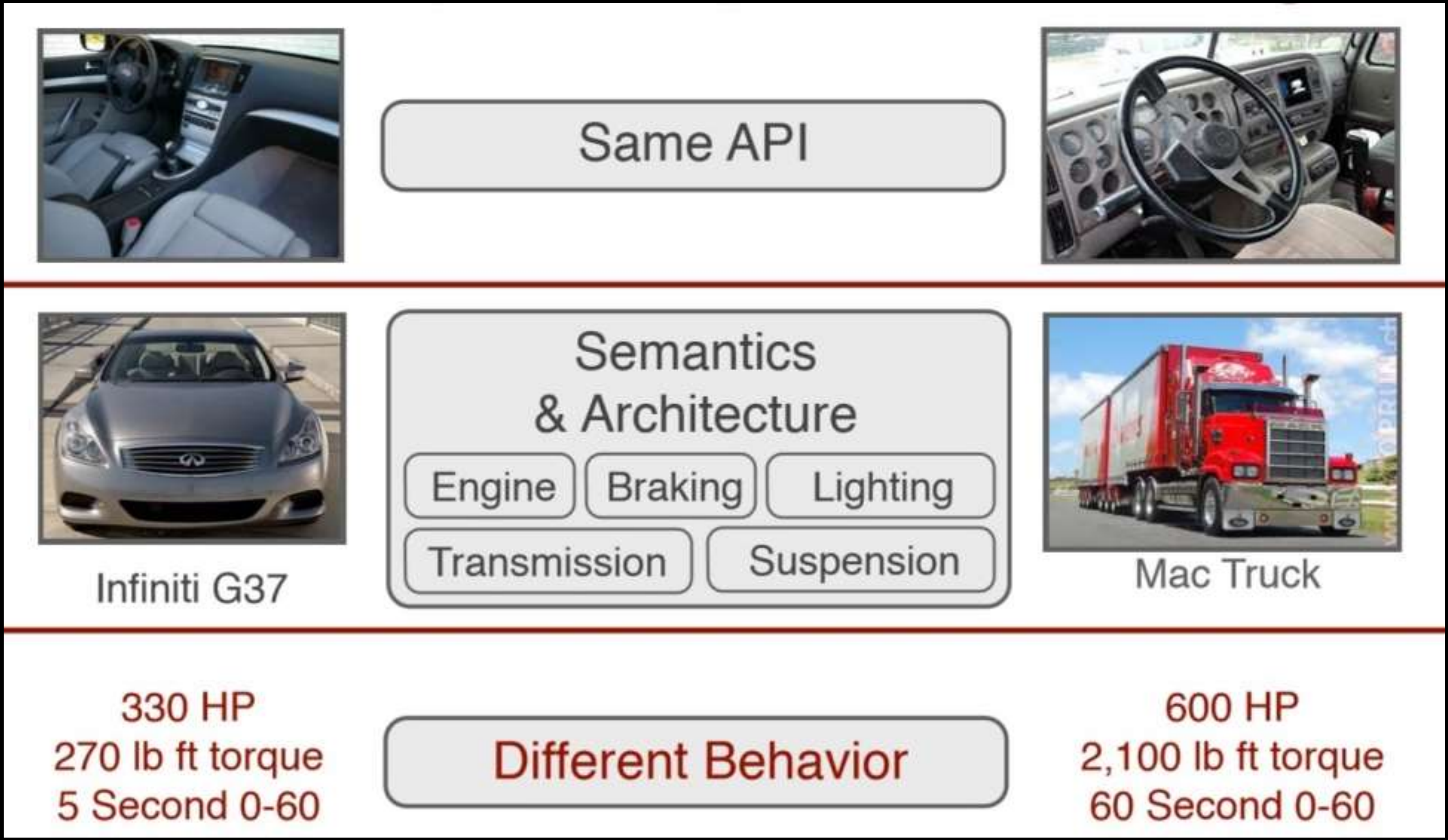


Four models for SQL on Hadoop

1. Parse and compile SQL into MapReduce jobs
2. Put a SQL interpreter on a generic execution engine
3. Run a native SQL engine in the cluster
4. Run a SQL interpreter on a non-generic engine (this will limit SQL functionality based on the underlying engine).



What’s under the hood matters in when querying



Same API

Semantics
& Architecture

Engine Braking Lighting
Transmission Suspension

Infiniti G37

Mac Truck

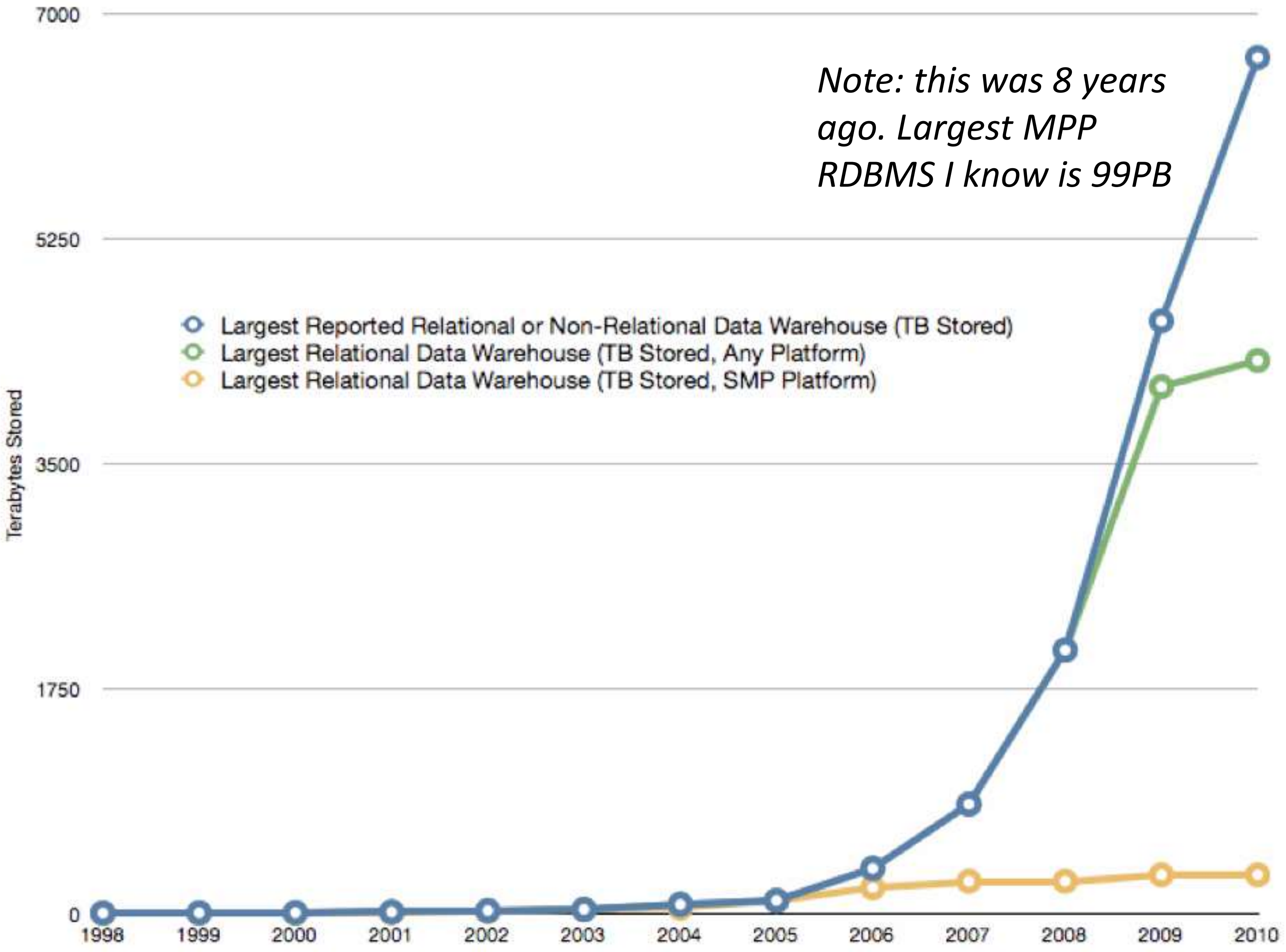
330 HP
270 lb ft torque
5 Second 0-60

Different Behavior

600 HP
2,100 lb ft torque
60 Second 0-60

Source: Randy Bias

The core problem of old BI was scalability. This is solved. New uses require new platforms for different workloads.



Note: this was 8 years ago. Largest MPP RDBMS I know is 99PB

Platforms

MPP DBs

Standard DBs

Source: Noumenal, Inc.

The shifting BI paradigm

The tool market is shifting, driven by new architectures that are enabled by new technologies.

Front-end tools are evolving away from BI-as-publishing, which changes their design, increases the burden on back end databases and creates new interaction challenges.

You need to evaluate tools based on more usage scenarios and interactive capabilities, less on report-building / dashboard features.

One standard tool is not the norm, it's the exception.

脳（あまり大きくない）
ここに耳がある
水中でも呼吸できる巨大な肺
ろっ骨
心臓
胃
ウラン袋
核反応袋（放射火炎を作ったり全身にエネルギーを送る）
2万トンの体重をささえる筋肉
え／南村喬之
太くて堅い足の骨

It's nice, but it'll never replace playing outside in the fresh air and getting plenty of exercise.
HistoricLOLS.com

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.

The right tool is the one that people will actually use, not the one you want them to use

"But we already have an enterprise standard"

About the Presenter

Mark Madsen is the global head of architecture at Teradata, Prior to that he was president of Third Nature, a research and consulting firm focused on analytics, data integration and data management. Mark is an award-winning author, architect and CTO whose work has been featured in numerous industry publications. Over the past ten years Mark received awards for his work from the American Productivity & Quality Center, TDWI, and the Smithsonian Institute. He is an international speaker, chairs several conferences, and is on the O'Reilly Strata program committee. For more information or to contact Mark, follow @markmadsen on Twitter or visit http://ThirdNature.net

Third Nature

About the Presenter

Shant Hovsepian is a cofounder and CTO of Arcadia Data, where he is responsible for the company’s long-term innovation and technical direction. Previously, Shant was a member of the engineering team at Teradata, which he joined through the acquisition of Aster Data. Shant interned at Google, where he worked on optimizing the AdWords database, and was a graduate student in computer science at UCLA. He is the coauthor of publications in the areas of modular database design and high-performance storage systems.
Follow him @superdupershant

ARCADIA DATA