#CONTEXT

Josh

@jherritz

Jordan

@jordanbarrette

Hi we're Josh and Jordan

#CONTEXT

MIOsoft®

And we work for MIOsoft. We're here to give you a peek at some of the R&D MIOsoft has been doing since 1998. .

#CONTEXT

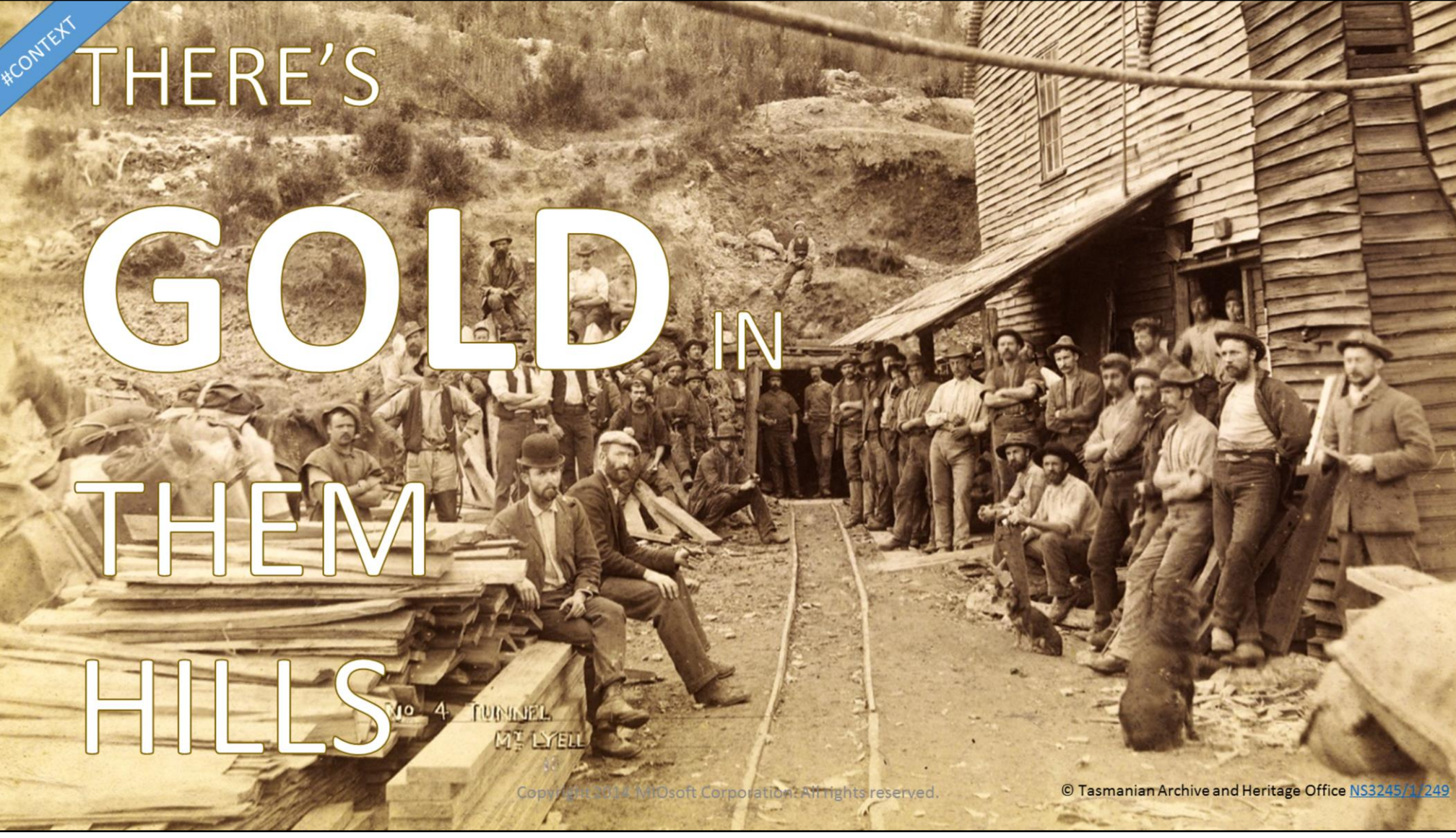
MIOsoft has its world and R&D headquarters in Madison, and major offices in Hamburg and Beijing. We started in 1998 with a project for CUNA Mutual Group.

When you ask us what MIOsoft does, we typically throw around a lot of technical terms. Or buzz words.

But really, what we're focused on is data.

And it's everywhere. In business systems. In sensors. In machines.

Companies are just now starting to realize this. And they are trying to gather it up as cheap as possible, hoping to derive some insight or discover some new business line.



Its like the gold rush. Everyone's digging in their data for something.

The real value of the data is what sorts of relationships it can show you as a business. You want to find relationships between pieces of data so you can contextually understand entities like customers. You want to find statistical relationships between business outcomes and contextual state, so you can influence future business outcomes.

From the business side, they're looking for a silver bullet. They want prescriptive analytics to enable proactive service. For a customer-centric business, you really want to treat every individual event or encounter in context of the customer. For instance, in a coffee shop the barista can be advised to automatically start making their favorite coffee, or a notification can be sent to the customer's phone with a list of their favorite items sorted from most likely to least likely with an offer to start putting the order together and to pay without having to wait in line. This would also be an opportunity to introduce the customer to something new based on discoveries made by observing other similar customers.

#CONTEXT

"Context is **one of the most important concepts** to application architecture"

In many ways, this is what we mean by contextual understanding. We call it context.

#CONTEXT

"Without contextual systems we will never get to the **nirvana** of what we envision around **cognitive engagement**"

Context is incredibly valuable, and analysts are starting to recognize that.

#CONTEXT

FRAUD PREVENTION
RISK MITIGATION
INTELLIGENCE
SERVICE PERSONALIZATION
SECURITY
COMPLIANCE
CUSTOMER 360-DEGREE VIEW
INVENTORY / SUPPLY CHAIN MANAGEMENT

When you can quickly discover and operate on contextually related data, ordinarily technologically hard development can instead be focused on business value and business use cases instead of architectural components.

From an architecture and development perspective, we define context as containing all the data, logic, and relationships you need to understand the state of an entity at a particular point in time, and provide relevant services such as concise views and proactive action

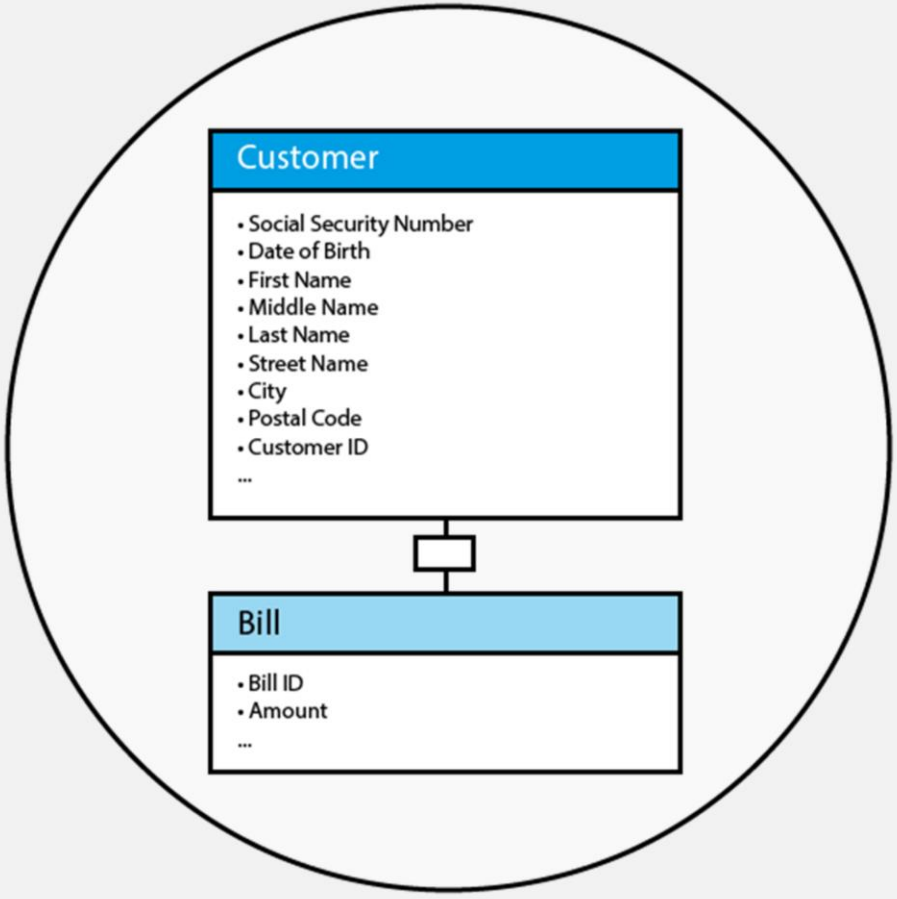
#CONTEXT

WE WANT TO

MODEL CONTEXT,

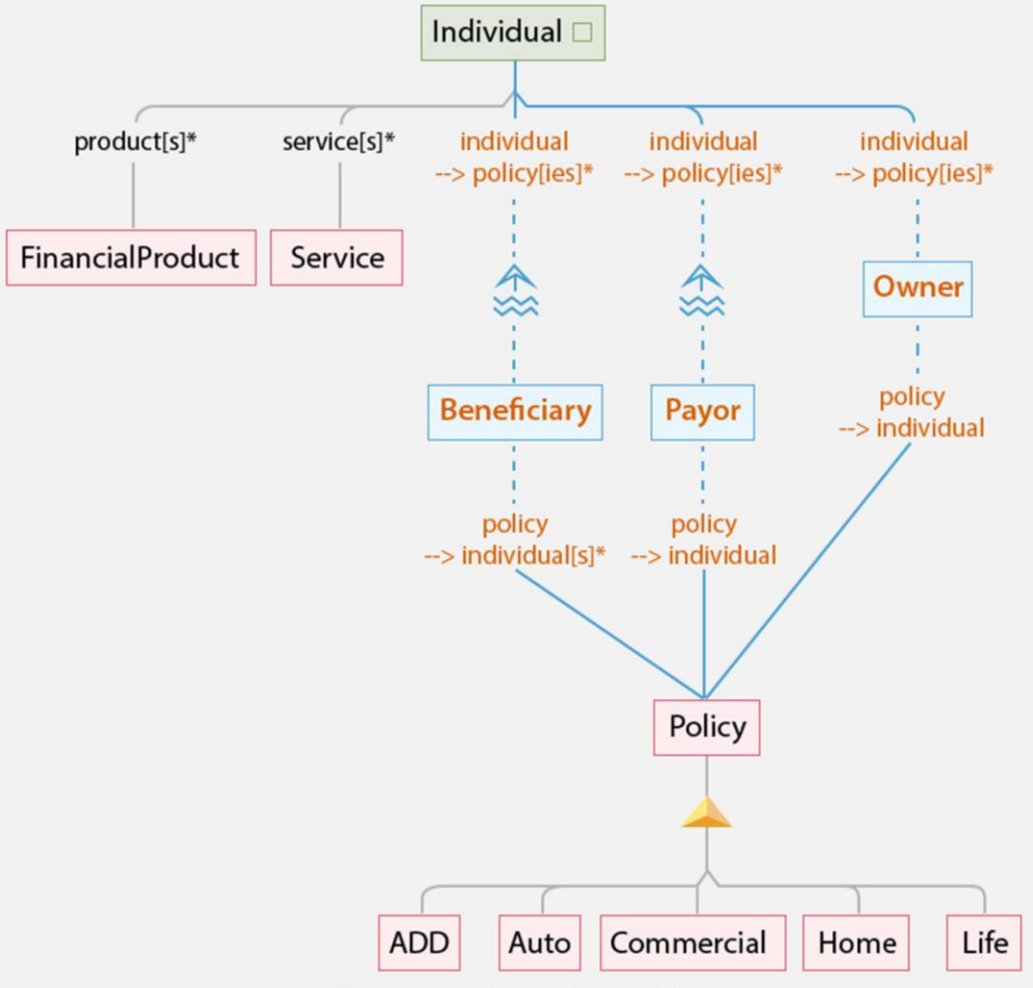
SERVE CONTEXT TO APPS AND LEVERAGE CONTEXT TO UNDERSTAND PATTERNS ACROSS OUR DATA.

#CONTEXT



Rather than modelling *structurally related* data like in tables, we want to model *contextually related* data together. We call it a context: a graph of closely related objects, that can additionally have relationships to other contexts.

#CONTEXT



A context model is essentially an object model with boundaries defined by relationships. This is convenient for OO programming. And like an object model, it includes behavior.

Context is exactly what we want to serve to applications, so they can leverage that unified understanding of an entity.

#CONTEXT

WE NEED TO

EFFICIENTLY STORE

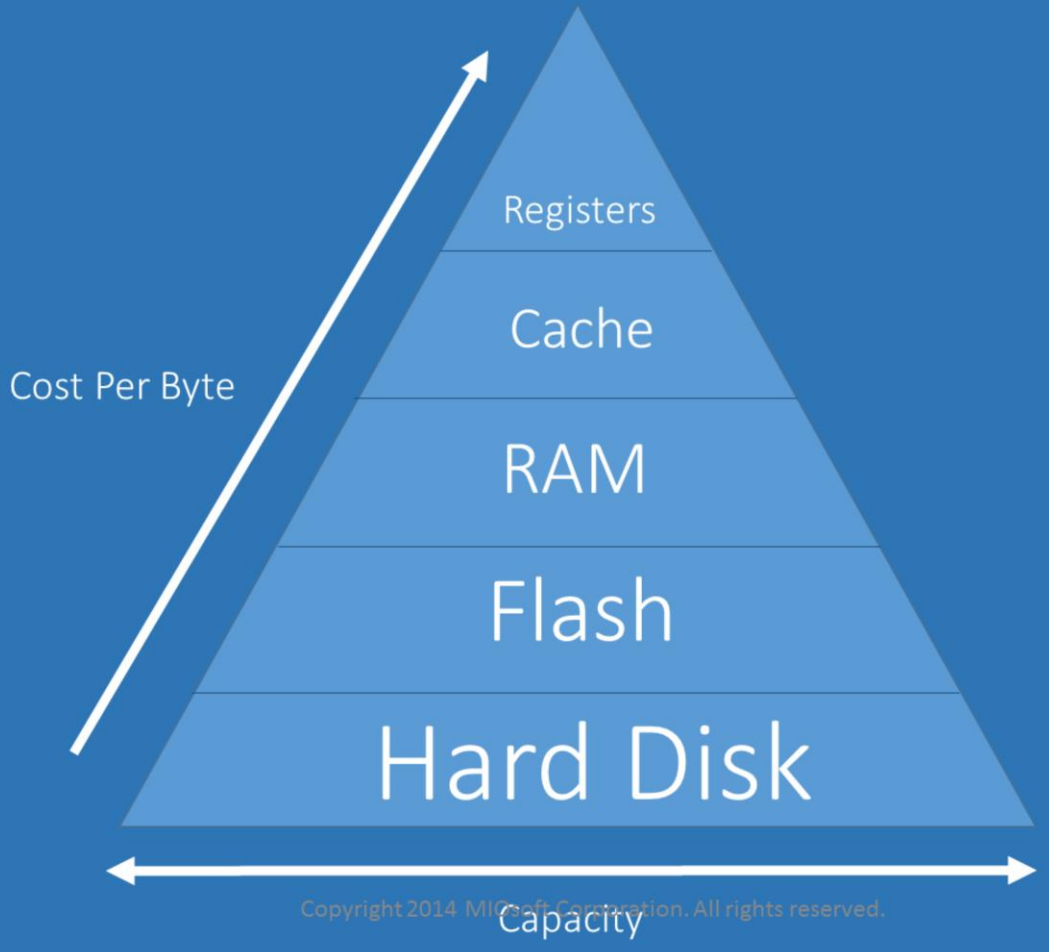
AND

QUICKLY RETRIEVE

CONTEXTUALLY RELATED DATA.

#CONTEXT

Computer Storage Hierarchy



We want hard disk because its high capacity and cheap.

#CONTEXT

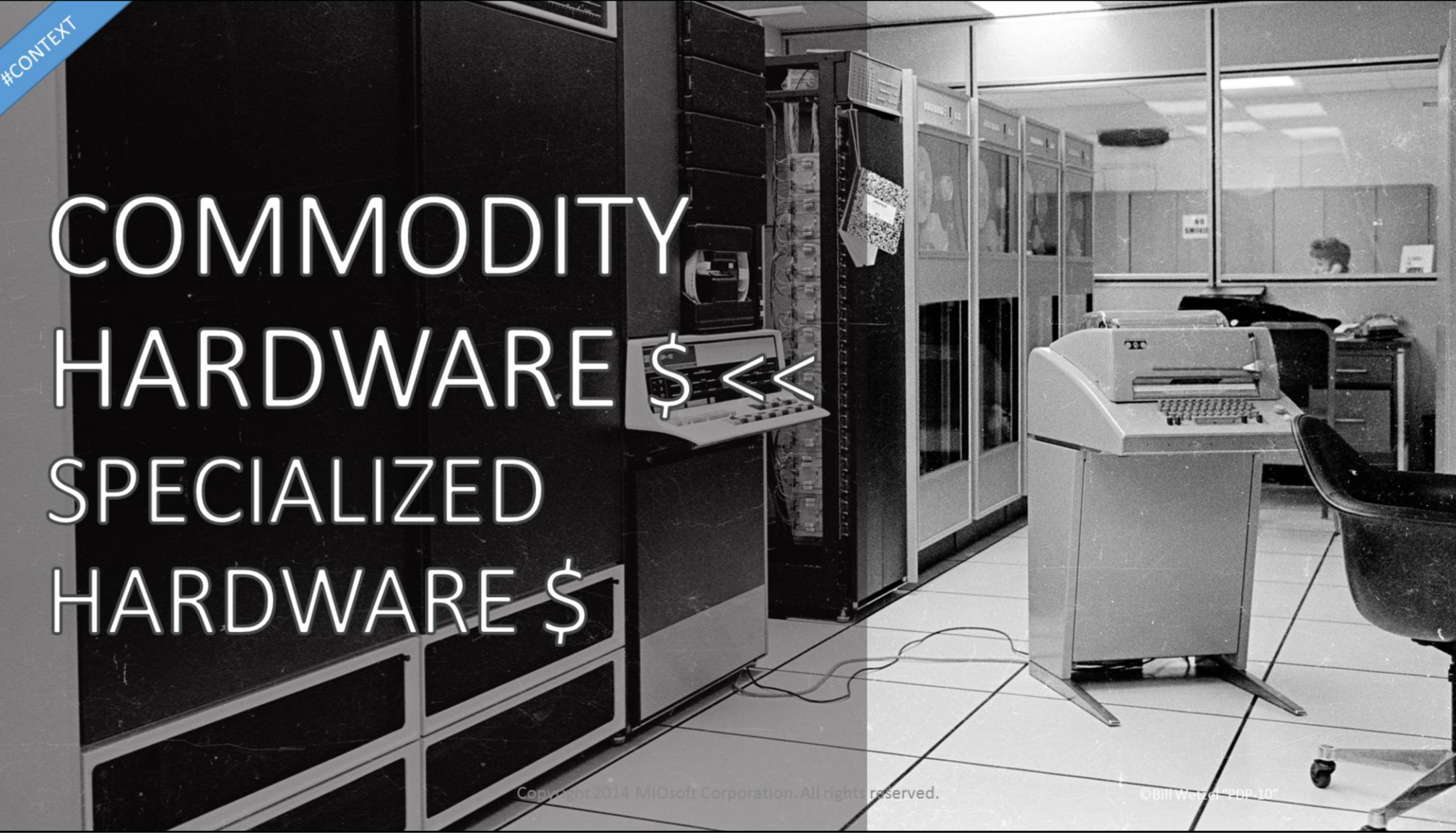
WE NEED TO

EFFICIENTLY STORE

AND

QUICKLY RETRIEVE

CONTEXTUALLY RELATED DATA.



And we know that commodity hardware is way cheaper than specialized hardware.

#CONTEXT

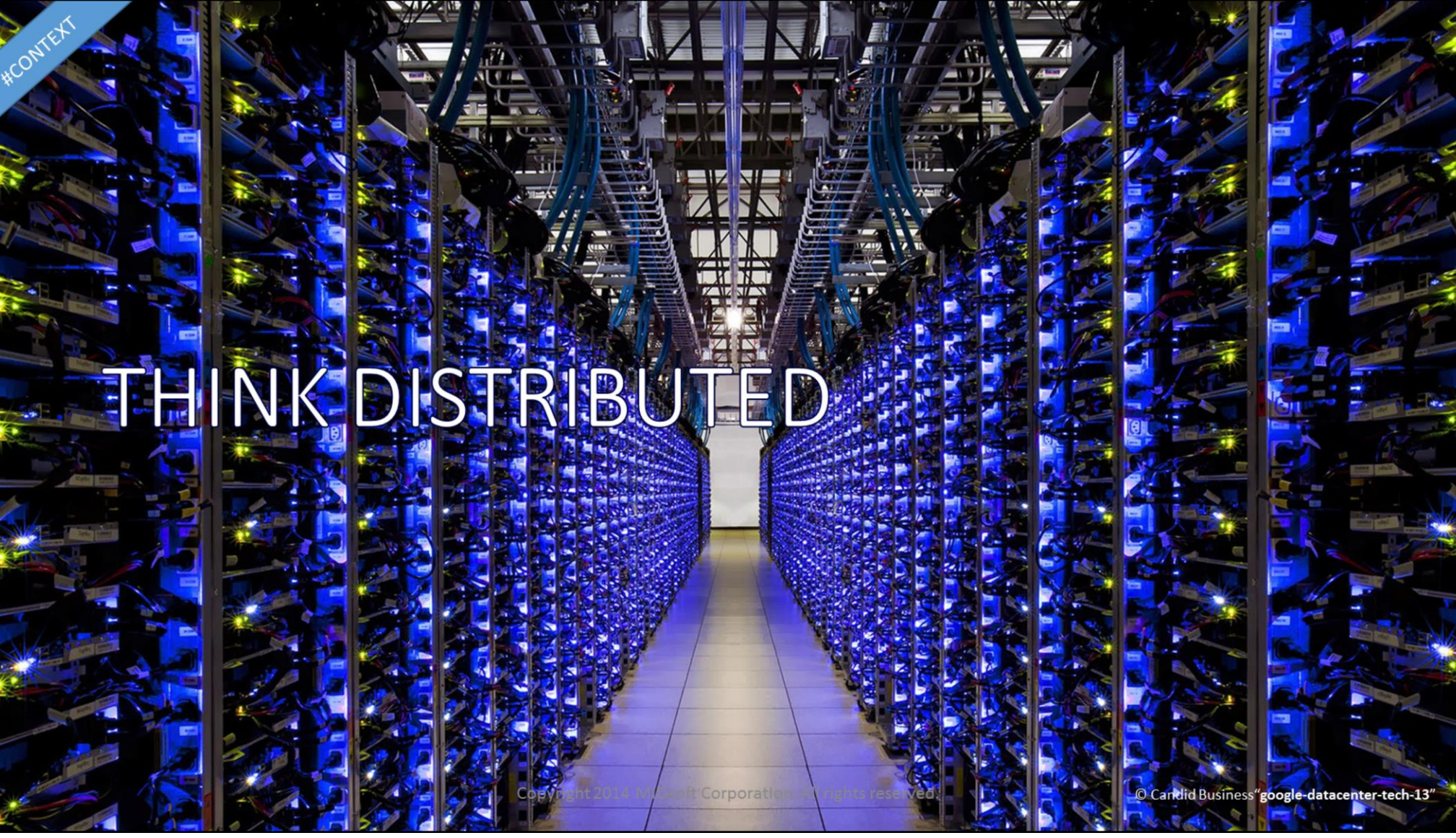
1 DELL R720XD = 48TBs

But 48TBs, although a lot of space, is hardly PBs.

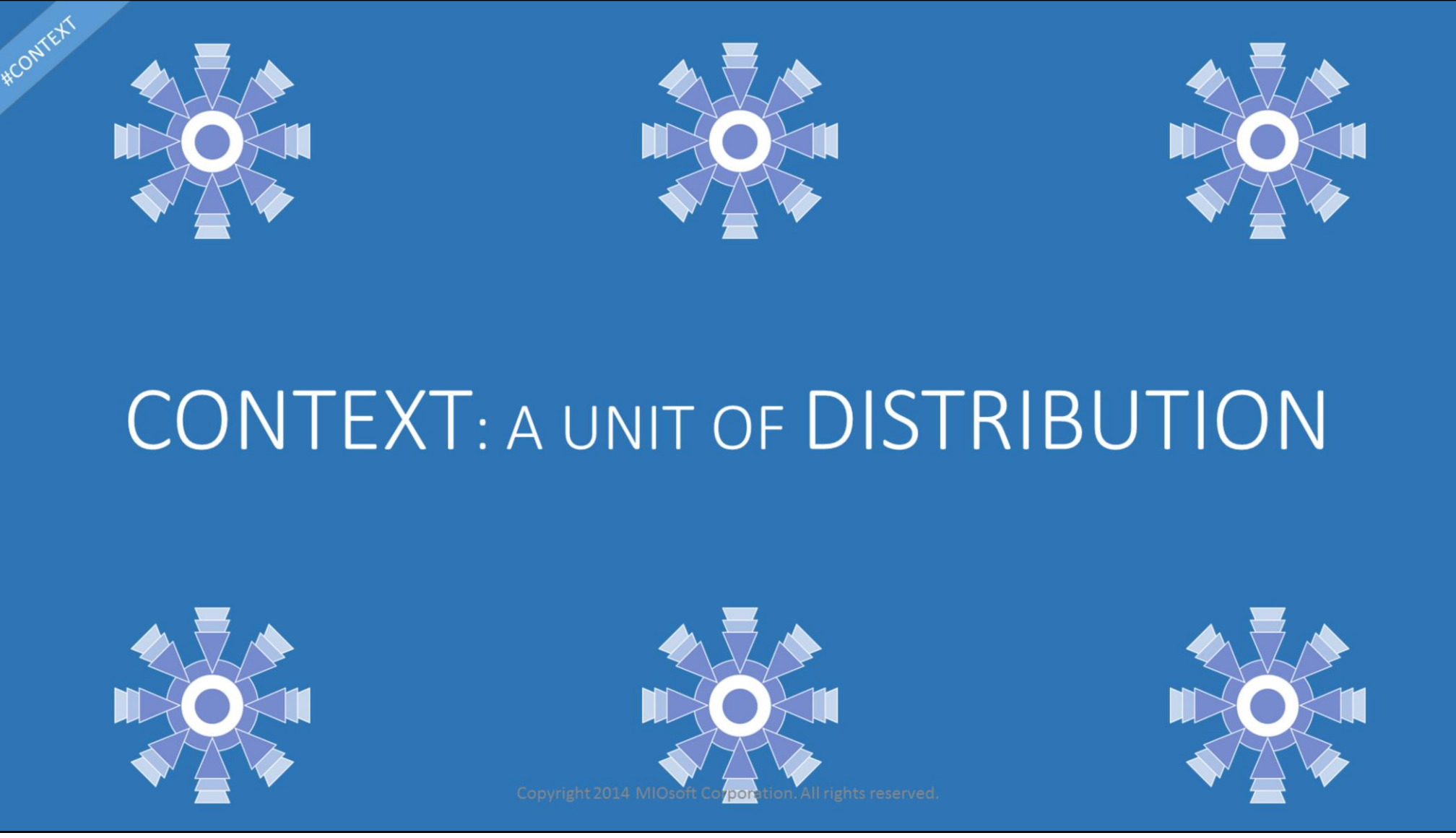
#CONTEXT

WE NEED TO
EFFICIENTLY STORE PBs+

This must be economical at scale. When WE SAY SCALE, THINK IOT+SOCIAL+TRADITIONAL. Think petabyes at the bottom end.



THINK DISTRIBUTED.



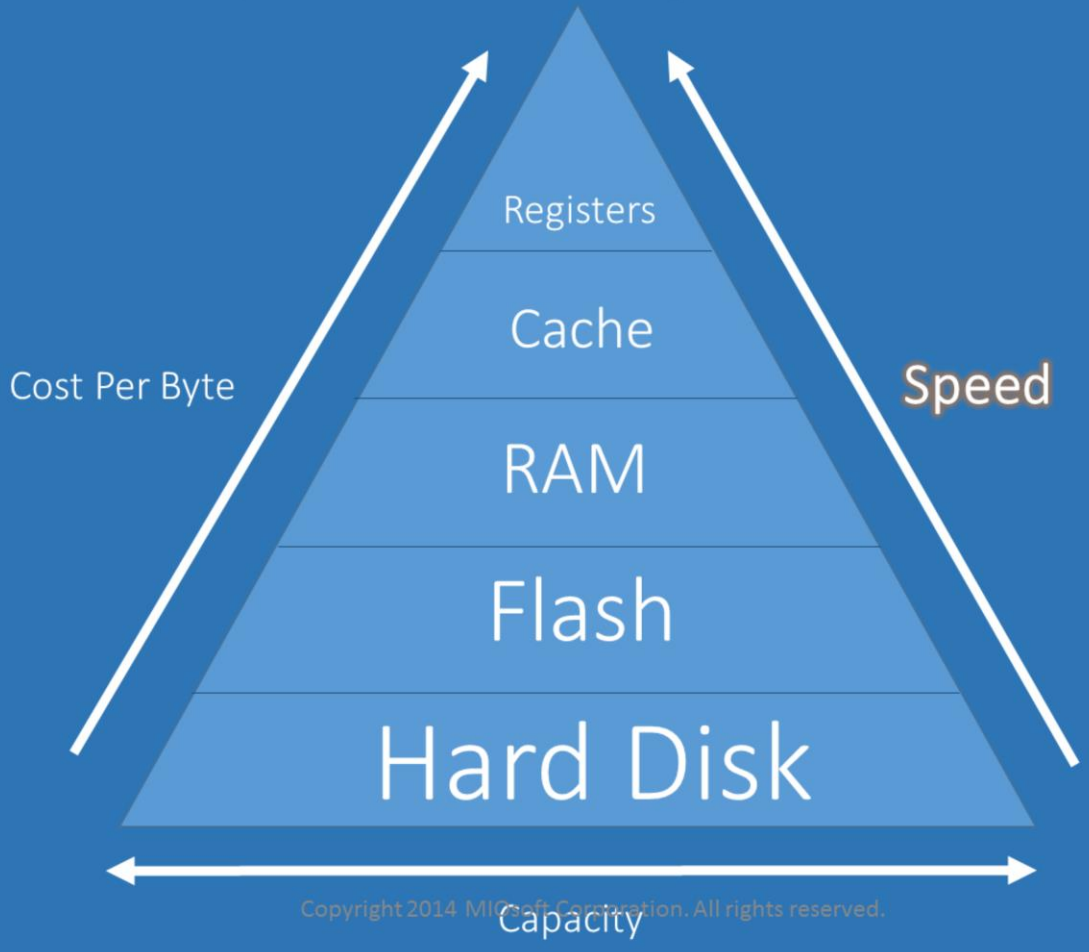
Lucky for us. Context can also be our unit of distribution. By spreading contexts across several servers, we can leverage the combined storage of commodity servers without sharding the database in an unnatural way like in relational. Since our contexts have behavior, **we also have a way to distribute processing**

#CONTEXT

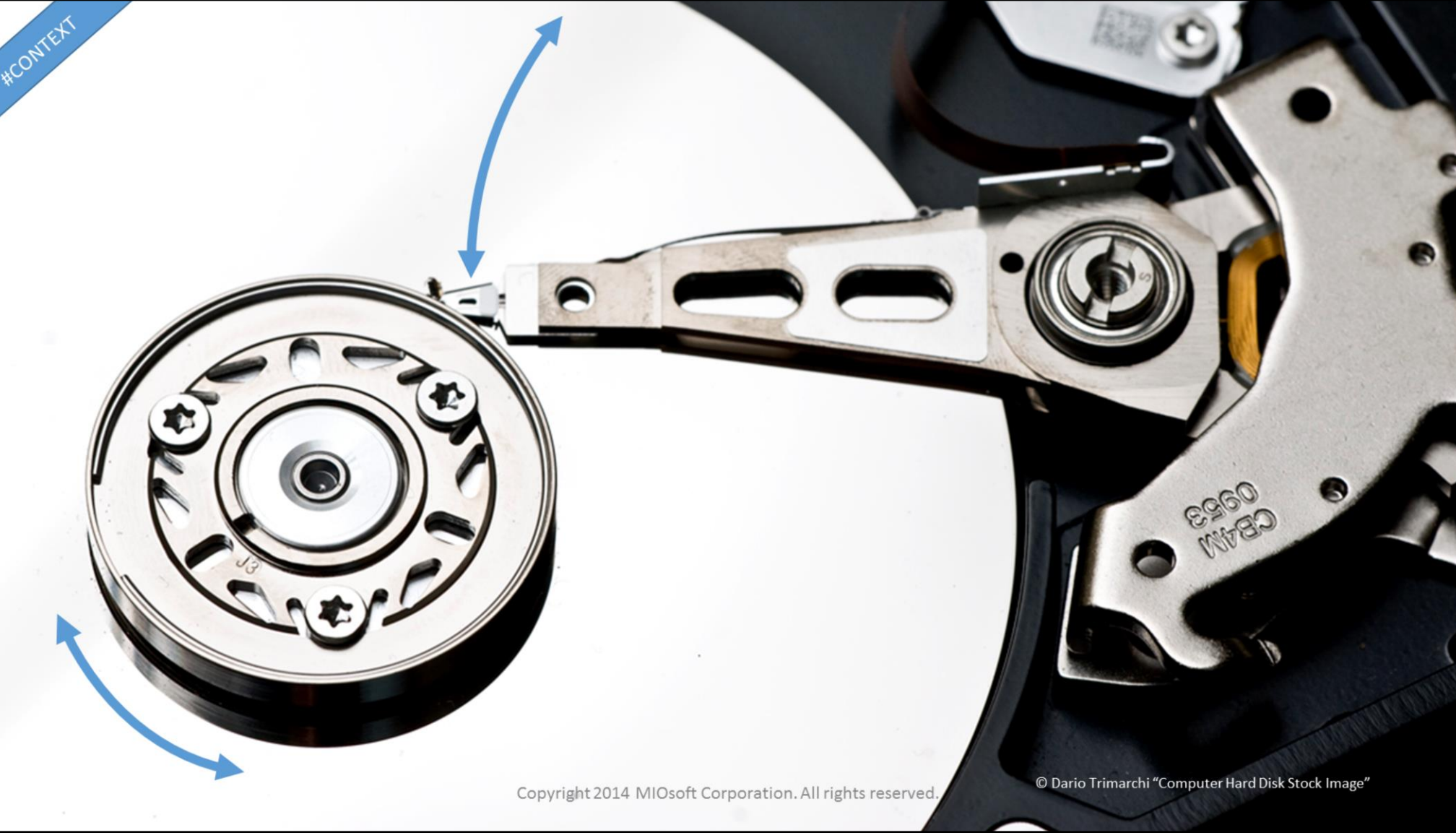
WE NEED TO
EFFICIENTLY STORE
AND
QUICKLY RETRIEVE
CONTEXTUALLY RELATED DATA.

#CONTEXT

Computer Storage Hierarchy



Like we said, we want hard disk because its high capacity and cheap. But its slow.



Its slow because of physics. The disk has to spin and head has to move physically to access a new region of the disk.

1 MILLION SEEKS = 83 MINUTES

1 BILLION SEEKS = 58 DAYS

How slow?

Say you want to assemble an application screen and it takes 100 queries (not out of line), at 5ms per seek that's a dedicated ½ second if nothing else is going on. Easily have to wait several second when multiple users are accessing.
Or say you want to do some ad-hoc analysis and aggregation on some sensor events. If each requires a seek, one million would take 83 minutes, one billion would take 58 days.

#CONTEXT

CONTEXT: A PHYSICAL AND LOGICAL CLUSTER OF RELATED DATA. READ/WRITE WITH ONE SEEK TO DISK.

Since we're logically clustering data anyway, why not just physically cluster it? That Customer-360-degree-view app now only needs 1 disk seek to get everything needed to populate a page.

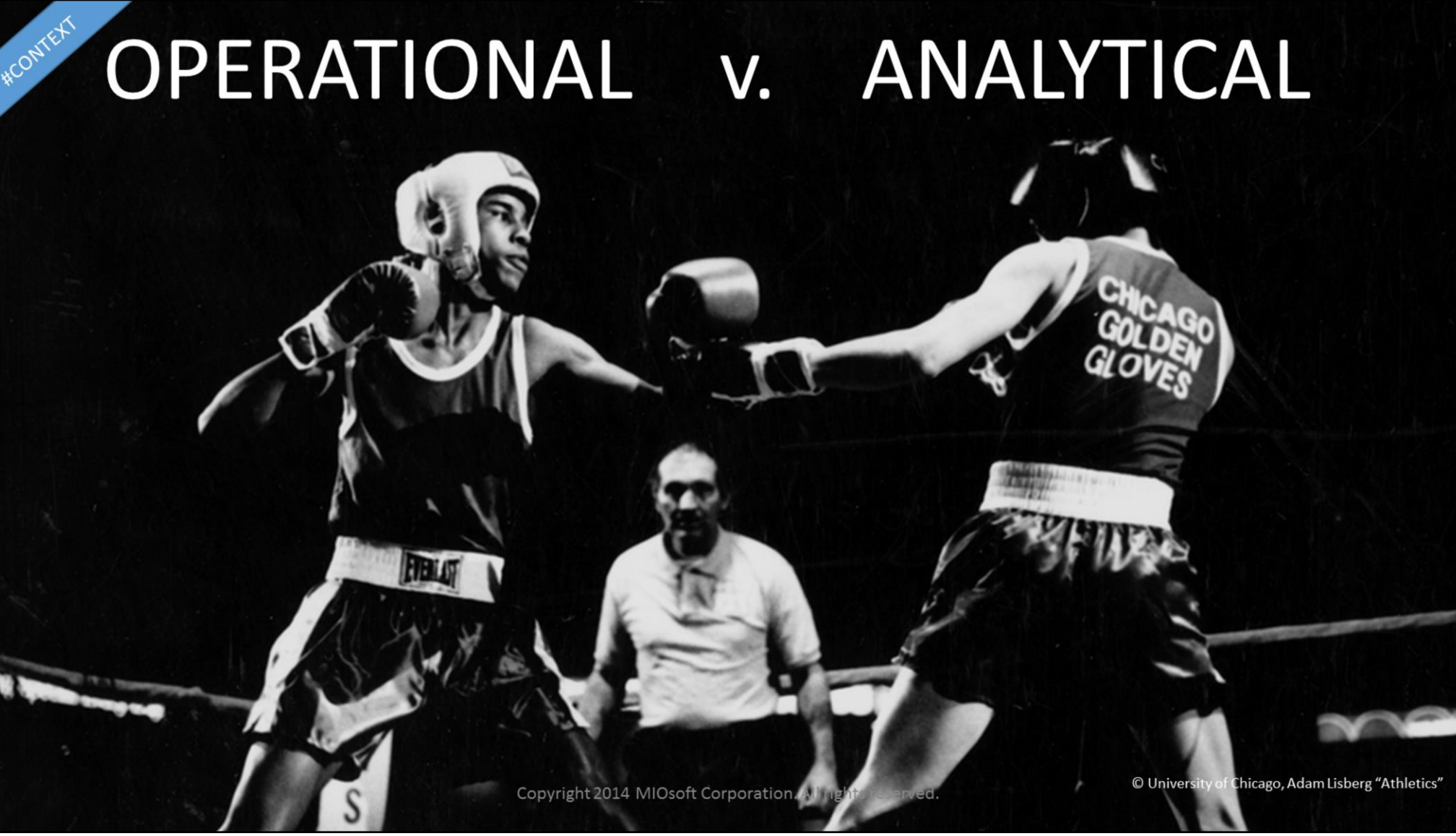
#CONTEXT

WE WANT TO
MODEL CONTEXT,
SERVE CONTEXT TO APPS AND
LEVERAGE CONTEXT TO
UNDERSTAND PATTERNS
ACROSS DATA.

At a very basic level, since each server is in charge of some number of contexts, and we are interested in potentially most of the contexts in the system, we can just have that server read in disk physical order to reduce seeks.

#CONTEXT

OPERATIONAL v. ANALYTICAL



Of course This is going to FIGHT our operational queries and updates. So how do we minimize contention between transactions, analytics, and queries

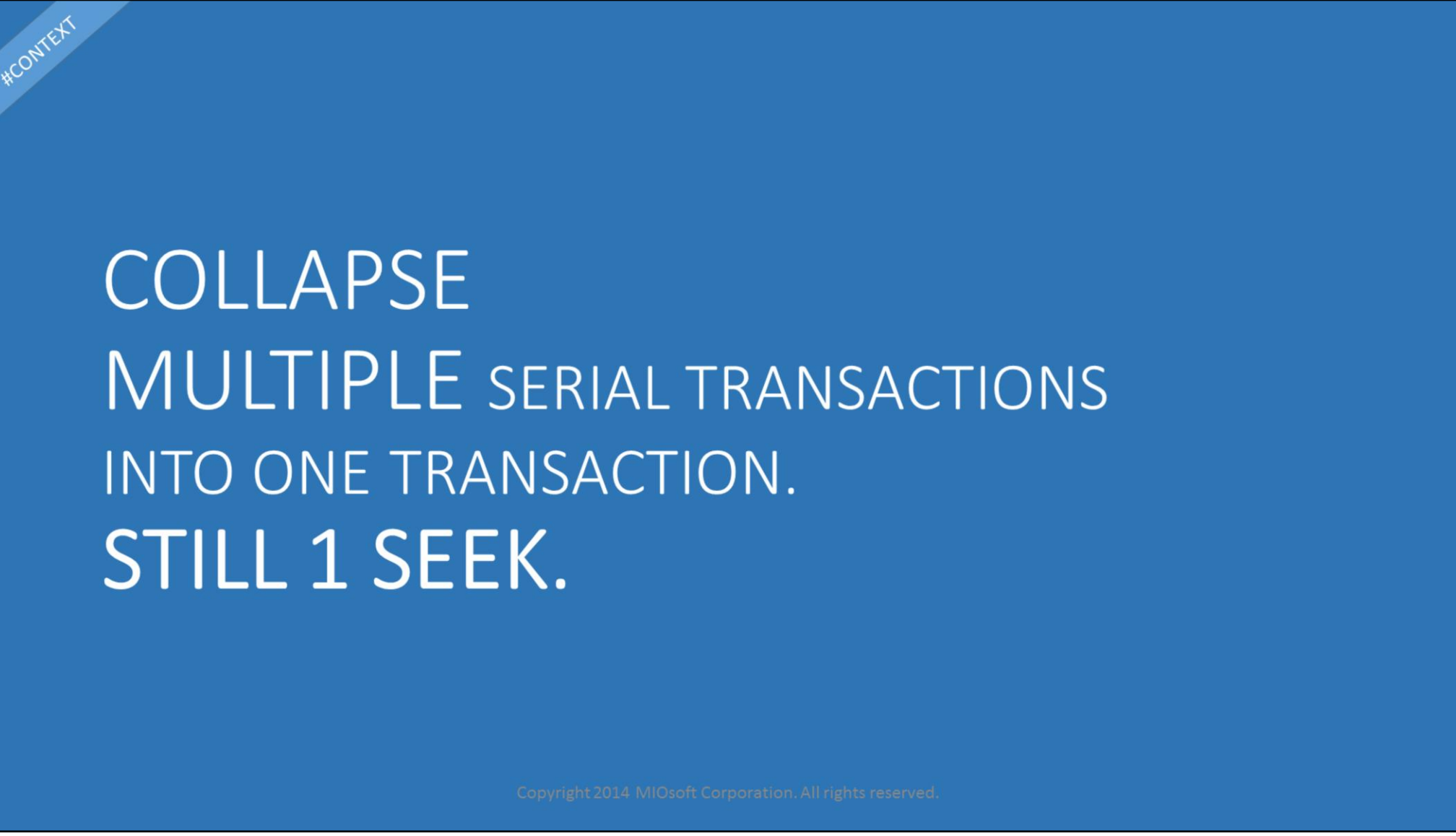
Most databases use write locking. Which usually means if something is writing nothing else can read. But even if we thought locks were great, maintaining locks on a large scale can cause a huge performance hit. We could choose to lock an entire context at a time to reduce individual object locks.

But we're still going to be wasting valuable time coordinating who goes next. And arbitrarily telling requestors to wait.

So if we're going to let transaction, queries, and analytics run, we're going to be bold and get rid of locks all together.

We queue up transactions to operate one at a time on a context. When a transaction is committed to the queue it is guaranteed to run, and if the requestor stops listening at that point it's essentially eventually consistent. However, if the requestor wishes, they can wait until they get confirmation that the transaction actually ran.

So how can we prevent a backlog from getting out of control? Presumably if we keep stopping we'll just exacerbate the situation, by introducing extra wait time to everyone in the line.

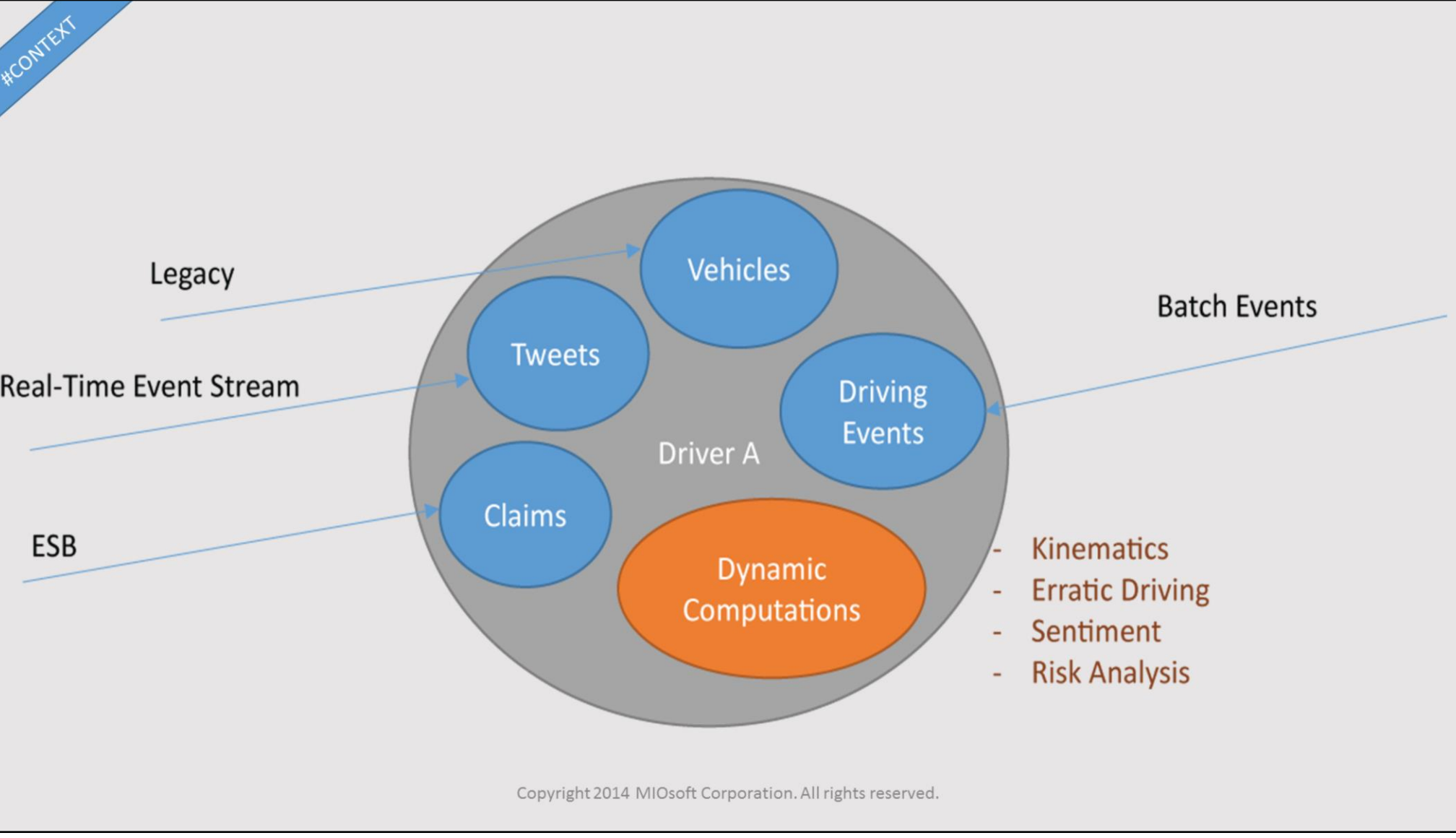


So instead, we actually use this stoppage to our benefit. We can batch up the transactions targeting a particular context and run them at the same time, with one write to disk.

#CONTEXT

NOW THE HARD PART

Now for the hard part.



How do we determine related objects from any source and cluster them in the first place? Turns out this is actually fairly difficult. As evident by the fact that the average amount of existing data enterprises analyze is only 12% according to Forrester Research. And that's only the data they already have.

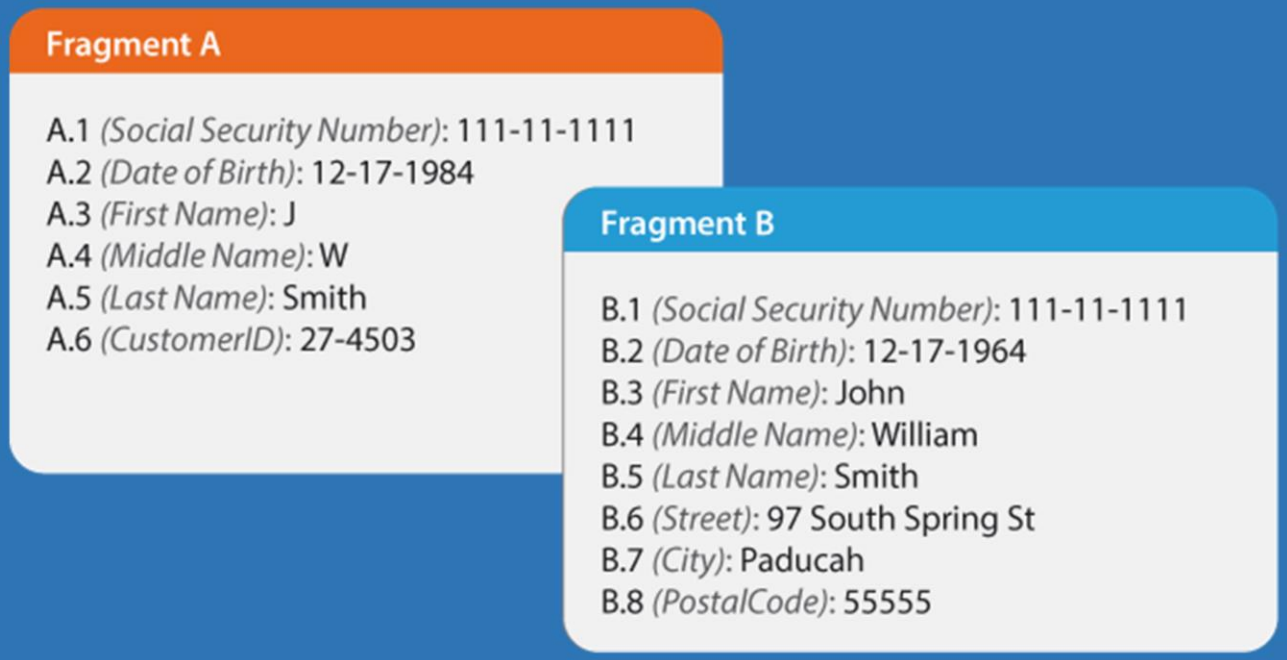
#CONTEXT

RELATIONSHIP DISCOVERY

What we want is to pour the data in and let the database figure it out. We want relationship discovery.


#CONTEXT
NEAR vs. FAR
RELATIONSHIPS
Copyright 2014 MIOsoft Corporation. All rights reserved.
© NASA, Visible Earch

When we say "NEAR" relationships, we mean data that belongs in the same context. If it's a customer context, for instance, then its about the same customer including supplementary data, like details about a customer's bills or perhaps recent GPS location. Near related data might include overlapping or conflicting pieces of information, even though there's theoretically a single contextual identity.

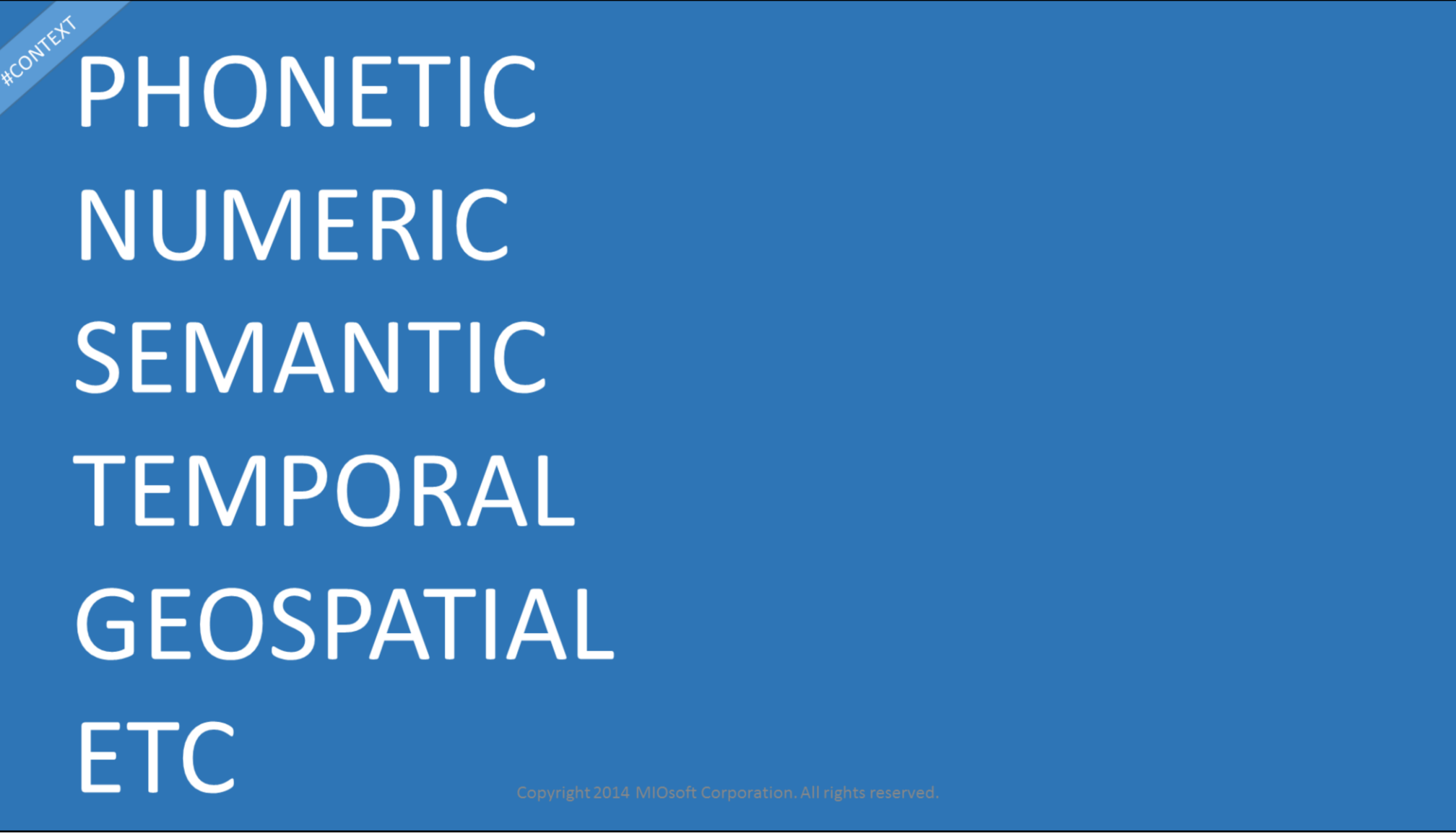


We call pieces of data that are incomplete representations of a context context fragments. The goal is to recognize when context fragments are related enough to put them in the same cluster.

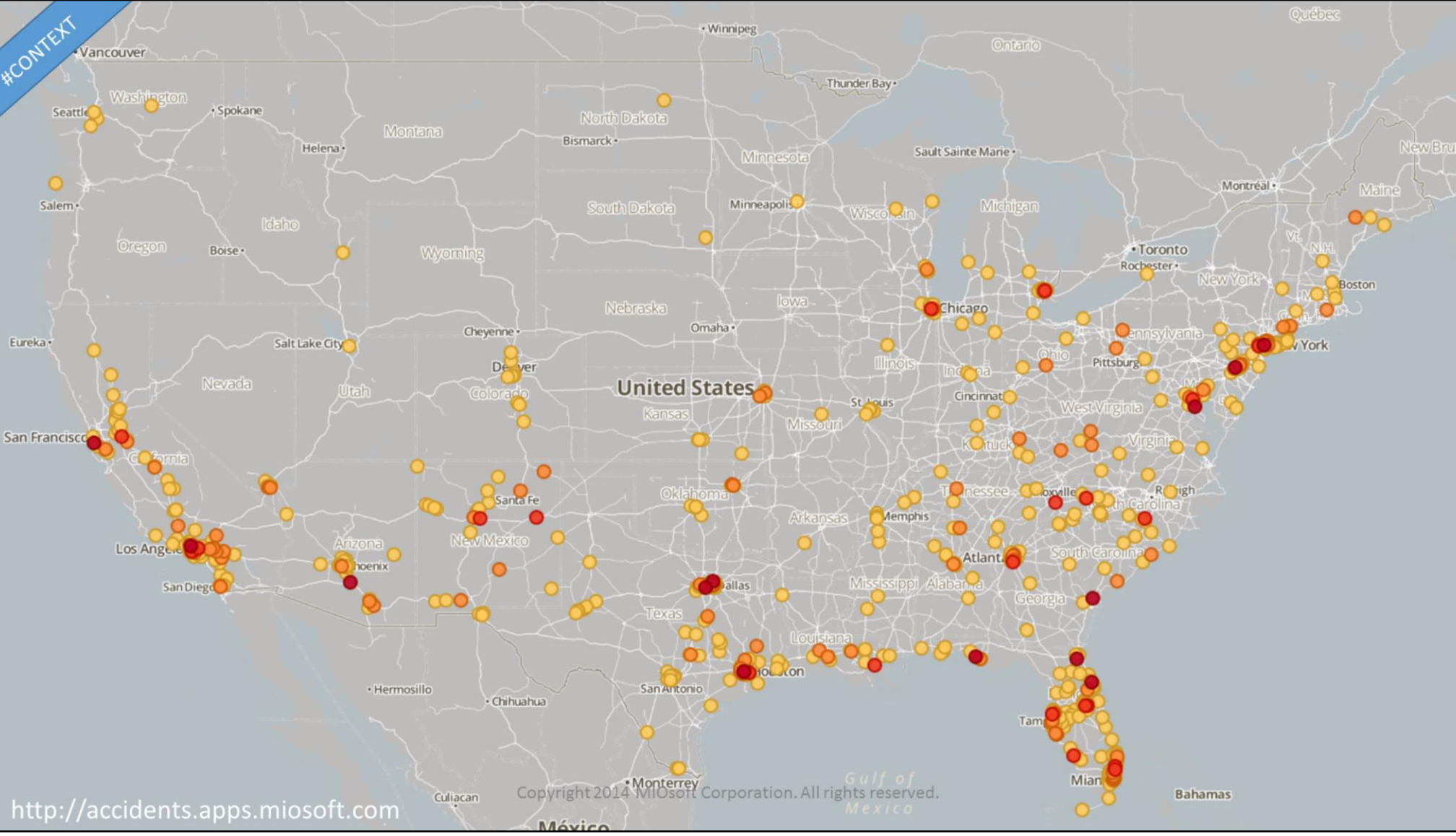
#CONTEXT

DISTANCE-BASED CLUSTERING

To do this we use a distance based unsupervised machine learning clustering engine.



There are many different ways you can define distance between two items. It could be geospatial, or even temporal or phonetic.

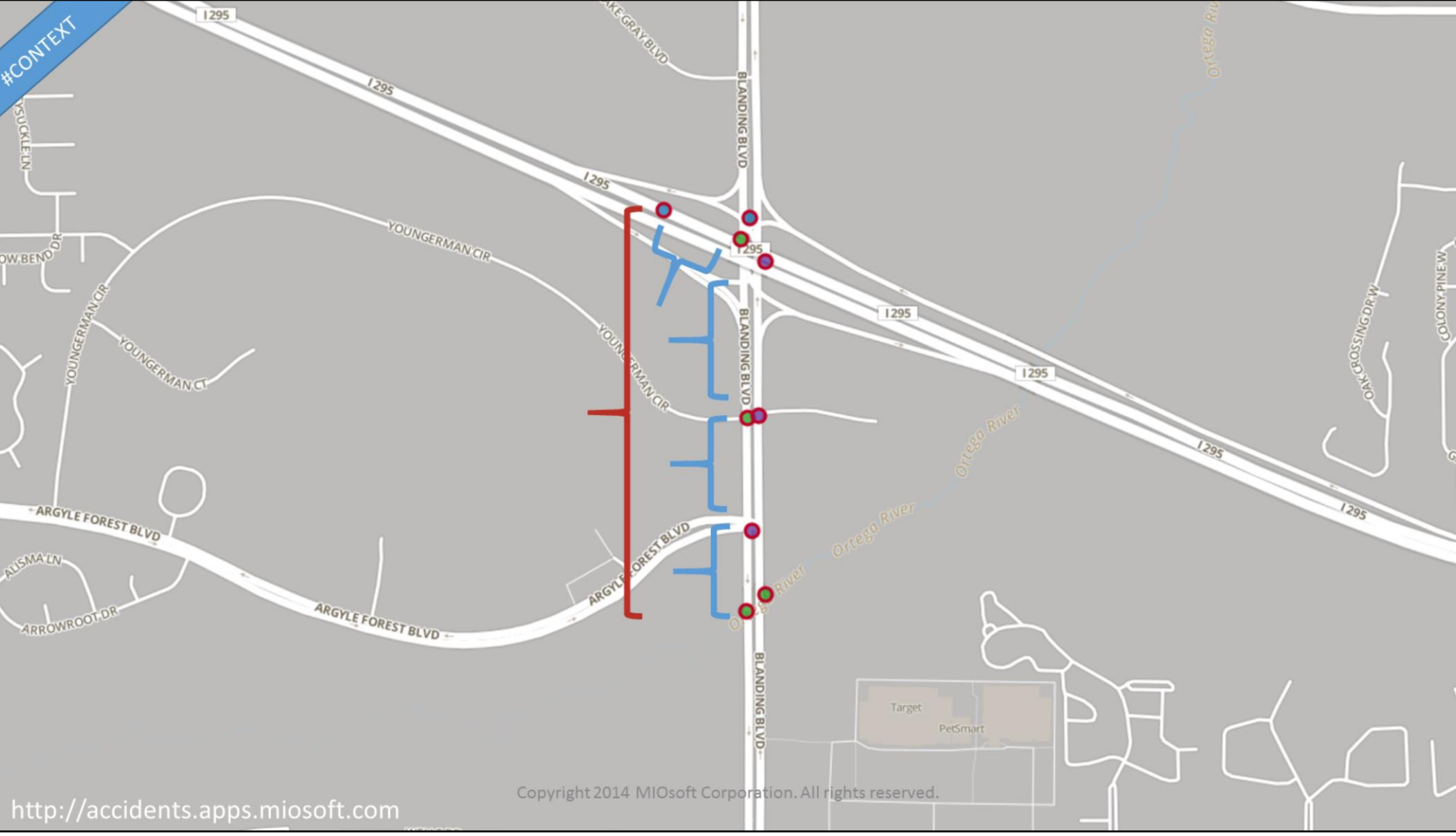


For instance, here's an example of distance-based clustering for fatal accidents to show clusters. It takes into account not only geospatial closeness, but also whether the accidents happened on the same street, or at an intersection.

#CONTEXT

TRANSITIVITY

There are a few concepts that make “NEAR” relationship discovery really powerful. First, we provide for context fragments to match transitively, meaning two fragments may be brought together if an intermediate piece of data is closely related to each.



For instance, in this example the top left crash is on a different street and too far away from the bottom-most crashes to match directly. However, it is linked through the intermediate crashes, that individually matched adjacent crashes in some way.

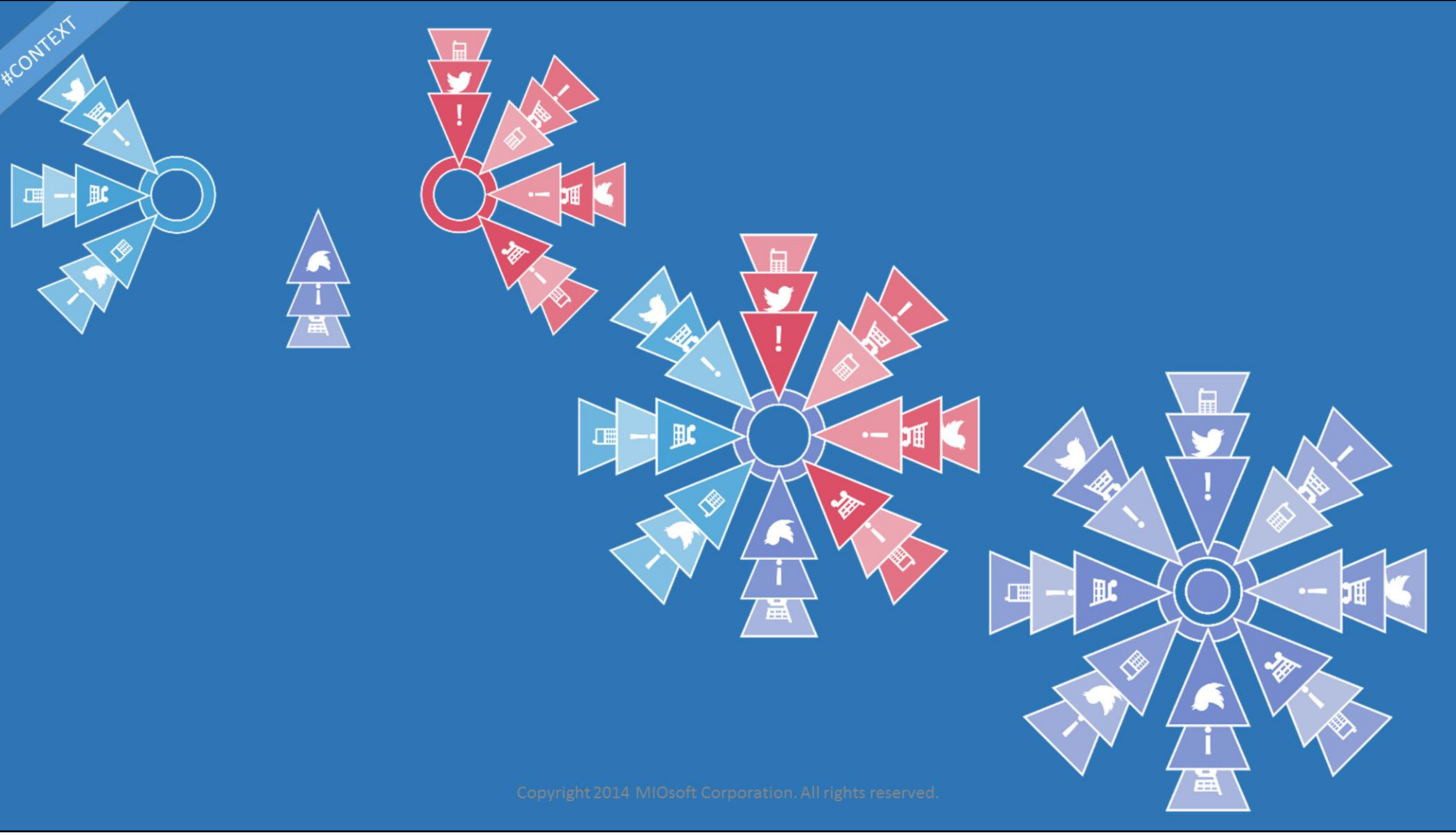
#CONTEXT

PROVENANCE

We also provide for provenance.

Provenance being where a fragment came from and how it ended up in a certain Context.

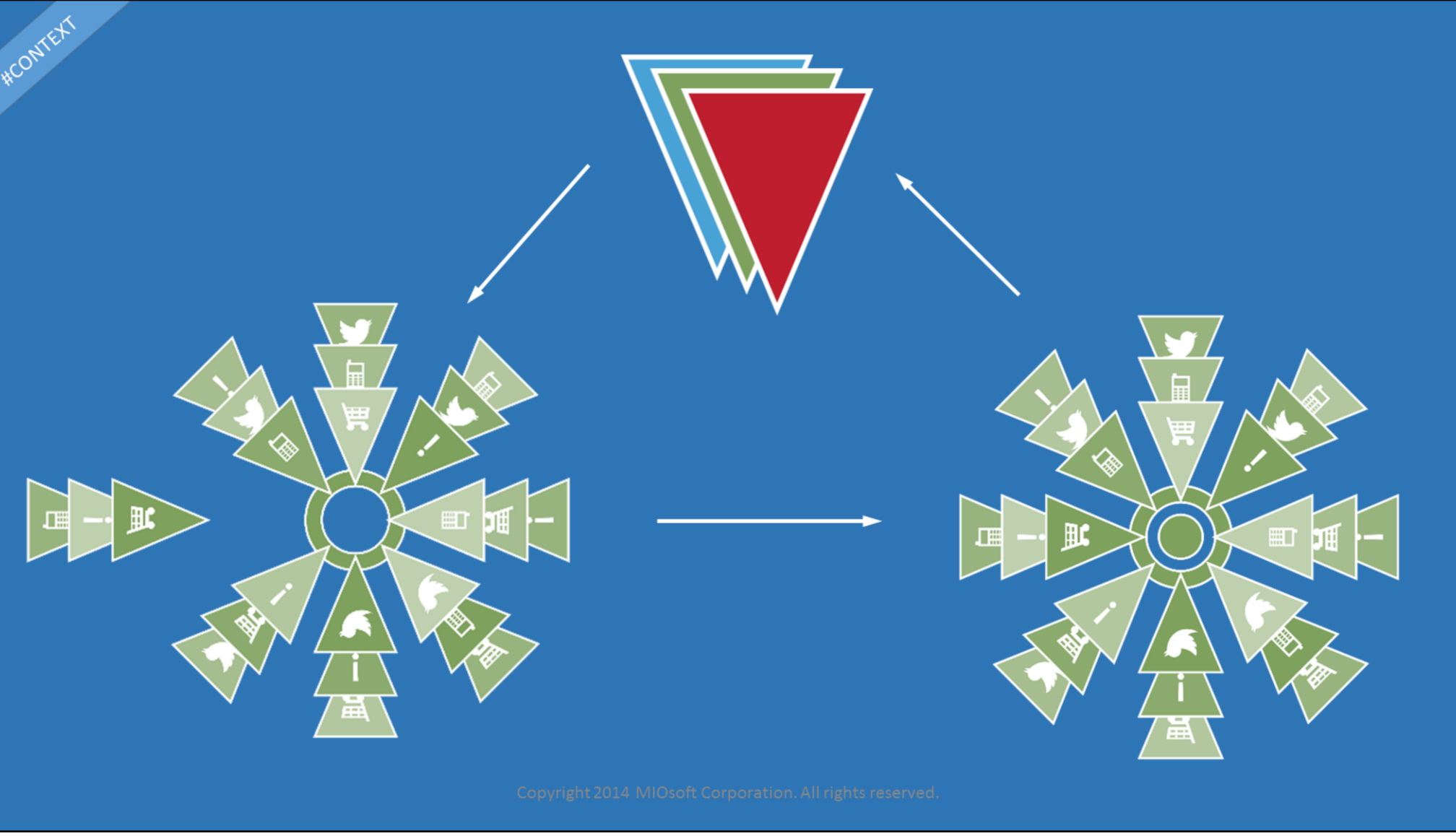
And provenance being the preservation of the fragment for the future in case its needed



Combining transitivity and provenance allows new fragments entering the system to cause contexts to merge or split, allowing the system to discover a better understanding of near relationships as new data is available.

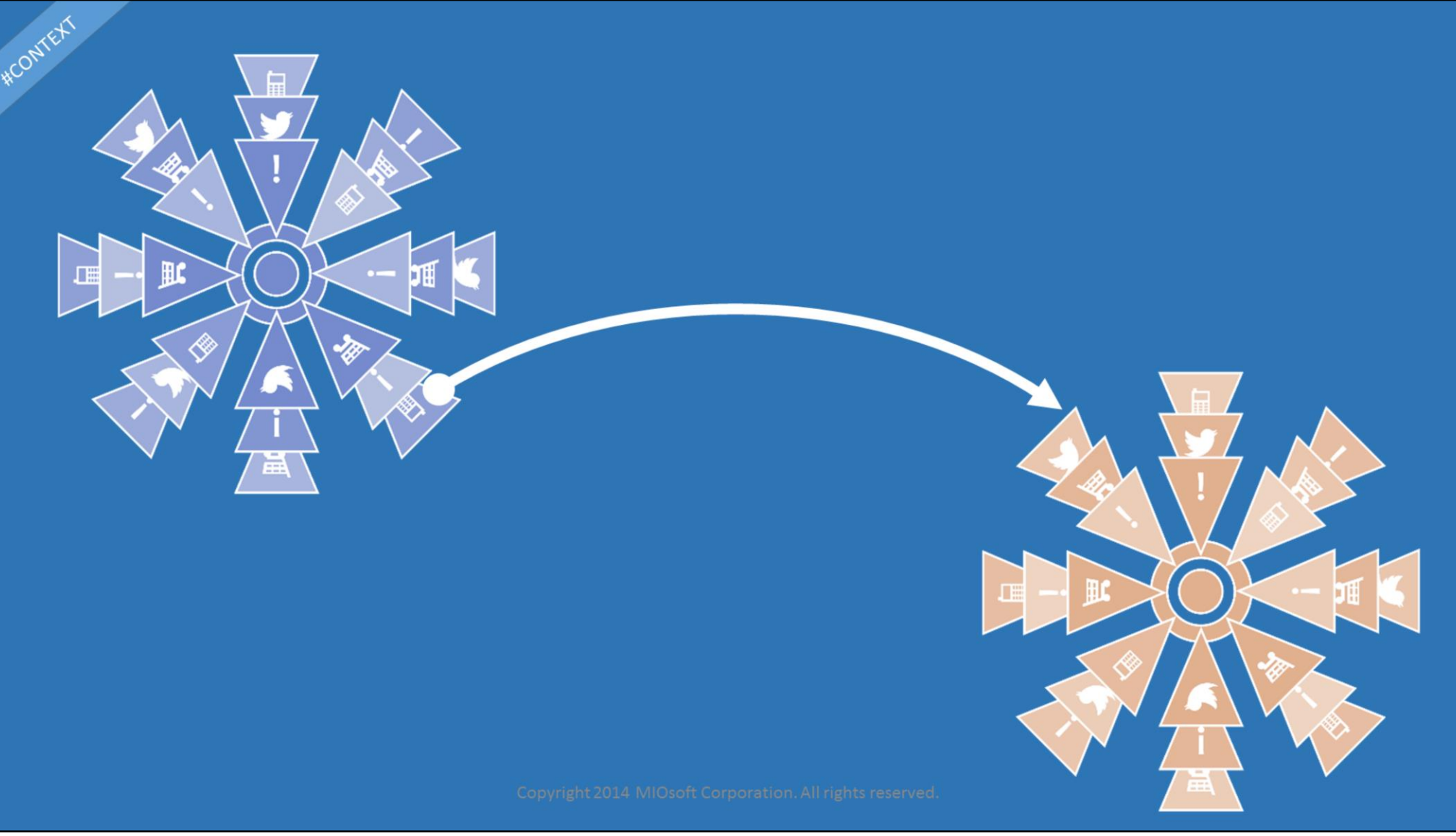
The data may also imply a relationship with another context. That’s a “far” relationship.

So when a context fragment is implying a relationship with another context, how do we find that context and build that relationship? Again the data might be incomplete or inconsistent. For instance, it could just be an identifier or a name.

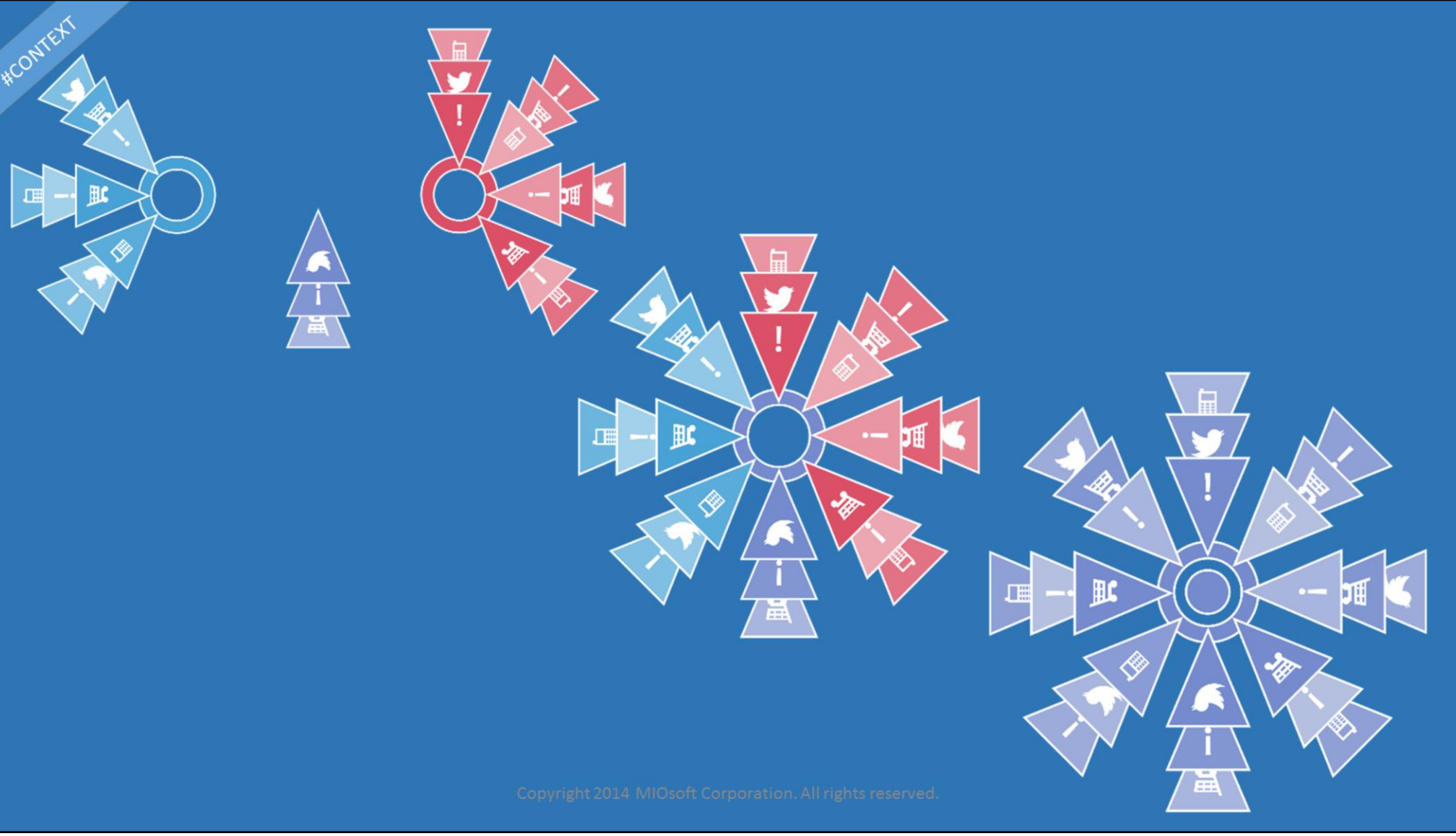


SO WE'RE GOING TO USE THE SAME METHOD. WE'RE GOING TO INJECT A SPECIAL FRAGMENT, CALLED A GRAPPLE FRAGMENT, WITH THE RELATIONSHIP DATA. THAT WAY, IT CAN USE THE SAME CONTEXTUALIZATION PROCESS TO FIND THE MOST APPROPRIATE CONTEXT, IF ONE IS AVAILABLE.

When it reaches a context, it will grapple back to the originating related context, and a relationship will be formed.



Relationships are first class citizens in our database model, so immediately both contexts will be aware of their new relationship. Think of intelligence analysis. Connecting the dots.



Since we use the same mechanism, we get the benefits of transitivity and provenance for relationships. For instance, if a new fragment arrives about an entity where only a grapple fragment existed previously, that context will be created and the related context will be grappled to automatically. In fact, the grapple fragment itself can cause contextualization of data, like context merging.

#CONTEXT

Questions.

#CONTEXT

ELASTICITY
COMMUNICATION
CONTAINERIZATION
CONFLICT RESOLUTION
RELATIONSHIP SUMMARY DATA
ADVANCED INDEXING
PARALLEL AGGREGATION
DATA CURATION TOOLS
OPERATIONS AND MANAGEMENT TOOLS
…

There's also a bunch of other pieces that are needed to make the system useful for mission-critical, large enterprise systems. Operations and management tools, elasticity, containerization, conflict resolution, automatic denormalization of summary data across relationships, advanced indexing, parallel aggregation, data curation tools, etc.

#CONTEXT

15+ YEARS IN THE MAKING

It’s a big project. It’s over 15 years of advanced R&D. And we’re now just really starting to talk about the technology beneath the business solutions.

www.miosoft.com