TAMING INTERNET-OF-THINGS

Exploiting IoT Analytics with Temporal SQL

+ richard hackathorn

Bolder Technology, Inc.

TABLE OF CONTENTS

CONTEXT

- Why is IoT Important?
- Diagnose & Predict Maintenance of Aircraft

01 PROBLEM

- Understanding All the IoT Data
- Analyzing Data from Flight Data Recorder
- Alignment & Gap Detection

02 SOLUTION

- It’s All About Time
- Temporal Normalization
- Temporal Intersection
- Boeing Data Pipeline

03 BENEFITS

- Compression & Efficiency
- Accuracy & Agility

04 FUTURE

- Geo-Spatial Tracking
- Systems That Never Break!

CONTEXT

WHY IS IoT IMPORTANT?

Internet-of-Things (IoT) has become a critical component in corporate information systems. Management at all levels should be aware of its possibilities and challenges.

This study highlights a specific technology – temporal SQL – as a key ingredient for efficiently analyzing large amounts of complex IoT data, generating practical dashboards for immediate action and datasets for advanced analytics.

At a deeper level, the key insight is to understand IoT behavior as being valid during a period-over-time, rather than an instance-in-time.

One can then determine whether specific combinations of behavior are valid by calculating precisely when time periods overlap.

An additional benefit is that the IoT data is compressed without information loss by factors of 100x, thus reducing processing times and storage requirements.

CONTEXT

DIAGNOSE & PREDICT MAINTENANCE OF AIRCRAFT

> "Pilot reported unusual sounds from left engine about an hour into flight.
>
> Lasted for 15 seconds.
>
> Normal flight otherwise.
>
> What happened?
>
> Should this aircraft be repaired before its next scheduled flight?"

A specific technique for using temporal SQL with sensor data was pioneered and patented by Ian Willson, Chief Architect Data & Technical Fellow, at the Boeing Company.

He applied it to diagnose and predict maintenance status of aircraft from the flight data recorder. This enabled a critical capability for assuring airline operations and ultimately company reputations.

+ WILLSON ET AL, SYSTEM AND METHOD FOR STORAGE AND ANALYSIS OF TIME-BASED DATA, US PATENT 2016/0125053, MAY 5, 2016.

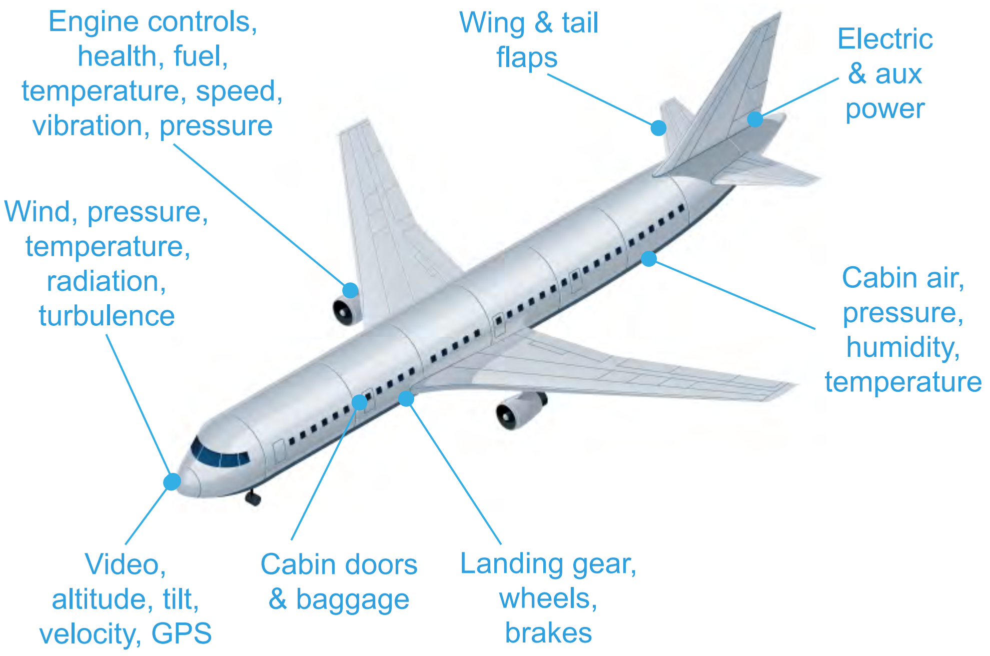
Problem

- Understanding All the IoT Data
- Analyzing Data from Flight Data Recorder
- Alignment & Gap Detection

01

PROBLEM
UNDERSTANDING ALL THE IoT DATA

IoT sensors generate lots of data. And it keeps on flowing!



Engine controls, health, fuel, temperature, speed, vibration, pressure

Wing & tail flaps

Electric & aux power

Wind, pressure, temperature, radiation, turbulence

Cabin air, pressure, humidity, temperature

Video, altitude, tilt, velocity, GPS

Cabin doors & baggage

Landing gear, wheels, brakes

Approximately 10,000 aircraft sensors

For each Boeing 787 flight, there are multiple data sources used to diagnosis airplane malfunctions.

For instance, there is:

- the pilot log (analyzed via text processing)
- engine maintenance faults (sent in-flight)
- various mechanical indicators (downloaded after landing).

The primary data source is the flight data recorder (FDR), the famous black box (it's actually orange). Service personnel manually download FDR data after each landing.

PROBLEM
ANALYZING DATA FROM FLIGHT DATA RECORDER

Over one year, there are 340,000 flights for just the 787 fleet across all airlines globally. Collecting data from all aircraft results in many petabytes of raw data and trillions of records.

Flight ID	Sensor ID	Time millisecond offset	Value
123456	123	15.385	45.89
123456	456	15.387	1.09
123456	789	15.390	2.67
- - - 60M more rows - - -			

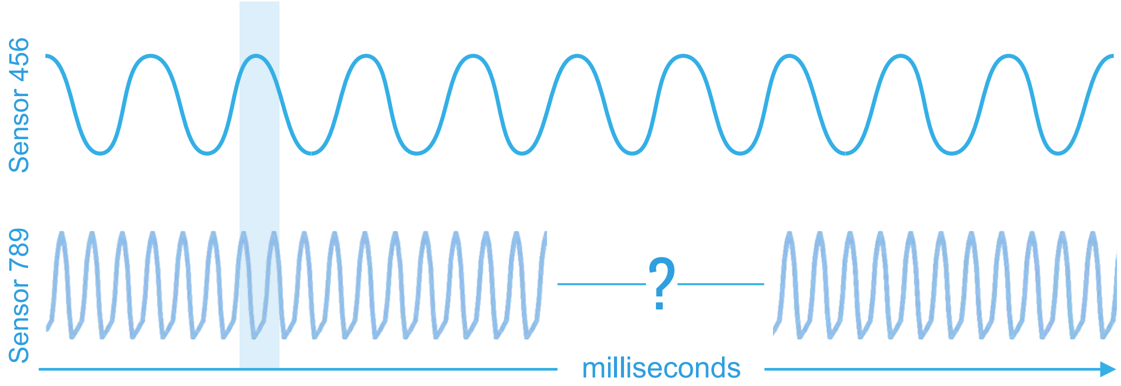
For a typical eight-hour Boeing 787 flight, the FDR data contains the recording from two thousand sensors, up to several times per second.

The resulting data is approximately 15 gigabytes in 60 million rows.

The above table is a simple example of FDR data. Time is the offset in seconds *from the start of the flight.*

PROBLEM
ALIGNMENT & GAP DETECTION

There are two key problems with analyzing FDR data – alignment and gap detection.



First, time alignment of sensor readings varies. Sensor readings are timed by the milliseconds (ms) from the flight start and sampled at different rates.

For example, one sensor may be sampled every second 502 milliseconds past the second while another is sampled every 2 seconds at 107 milliseconds past the second.

Second, there may be gaps in the sensor readings. For example, a sensor may send measurements every 100ms, skip several seconds, and then resume sending data. Identifying these gaps are often valuable in an analysis. For some sensors, gaps indicate a problem. For others, it's a normal condition.

Normal SQL is inadequate. Custom code required!

The use of basic SQL to process FDR data proved to be complicated, difficult to code, and inadequate.

Boeing had to develop complex, costly custom SQL that gave only approximate answers.

Further, one query analyzing one hundred flight hours **took 200 hours to execute!**

Yet, multiple such queries are needed to analyze a flight and then develop diagnostic machine learning models.

Solution

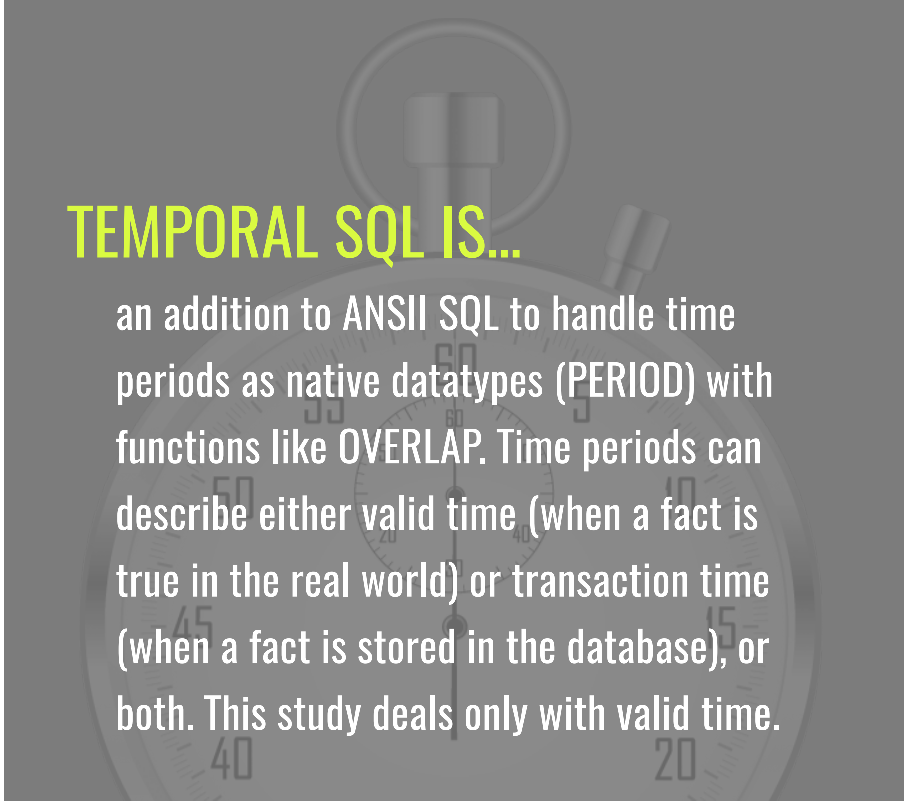
- It’s All About Time
- Temporal Normalization
- Temporal Intersection
- Boeing Data Pipeline

02

SOLUTION
IT'S ALL ABOUT TIME!

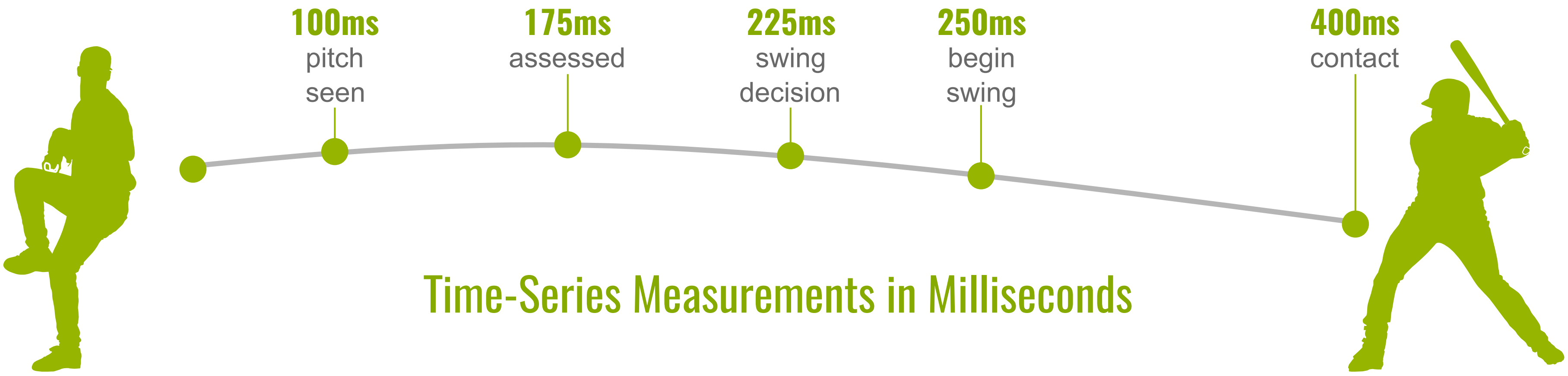
The solution developed by Willson and his team was to utilize the temporal SQL functions in Teradata® Database 14.10. It is an innovative two-stage process:

1. **Temporal normalization of the data-at-rest when loaded, followed by…**
2. **Temporal intersection during each query step – plus additional temporal normalization.**



TEMPORAL SQL IS…

an addition to ANSII SQL to handle time periods as native datatypes (PERIOD) with functions like OVERLAP. Time periods can describe either valid time (when a fact is true in the real world) or transaction time (when a fact is stored in the database), or both. This study deals only with valid time.



Time-Series Measurements in Milliseconds

SOLUTION
TEMPORAL NORMALIZATION

Flight ID	Sensor ID	Time Offset	Value
123456	789	15.390	2.67
123456	789	15.640	2.67
123456	789	15.890	2.67
- - - 100 similar rows - - -			
123456	789	41.140	2.67
123456	789	41.390	2.89

The first stage, temporal normalization, transforms sensor readings from an instant-in-time to a period-of-time. This period indicates the sensor value is the same between the start time and end time of the period during which there were no gaps in the readings.

Consider the tabular example for the values of a single sensor in chronological order.

...transforms sensor readings from an instant-in-time to a period-of-time.

About 15 seconds into a flight, sensor #789 has the value of 2.67 and retains this value for 26 seconds. *Since this sensor is sampled every 250ms, 104 rows are normally required* to indicate this fact.

continued

SOLUTION
TEMPORAL NORMALIZATION

temporal SQL

Below 104 rows are collapsed into one row showing the period during which 2.67 is the same value at every consecutive reading. Periods are intervals with a closed start but an open end.

Note that the PERIOD is a temporal SQL datatype and considered a single value. Hence, temporal SQL can process PERIOD values efficiently with functions to get/set start and end times, plus detect overlaps among periods.

The **alignment problem** is solved because the start and end times are maintained within the PERIOD datatype. There is no need to round time values for comparison if fine granularity is required.

The **gap deletion problem** is also solved by considering a period closed when a sensor reading is missing at its expected frequency. Hence, the normalized data will show exact gaps in the sensor readings.

Flight ID	Sensor ID	Period	Value	Rows Collapsed
123456	789	(2016-11-09 00:00:15.390, 2016-11-09 00:00:41.390)	2.67	104
123456	789	(2016-11-09 00:00:41.390, 2016-11-09 00:00:47.100)	null	
123456	789	(2016-11-09 00:00:47.100, 2016-11-09 00:01:58.489)	2.89	44

Note that the normalized period starts at 15.390 when the value becomes 2.67 and ends at (but not including) 47.100 when the value changes to 2.89.

BUT WAIT!!!
A sensor reading was expected at 41.390 and remained missing until 47.100. So, a "gap" reading with null value was inserted.

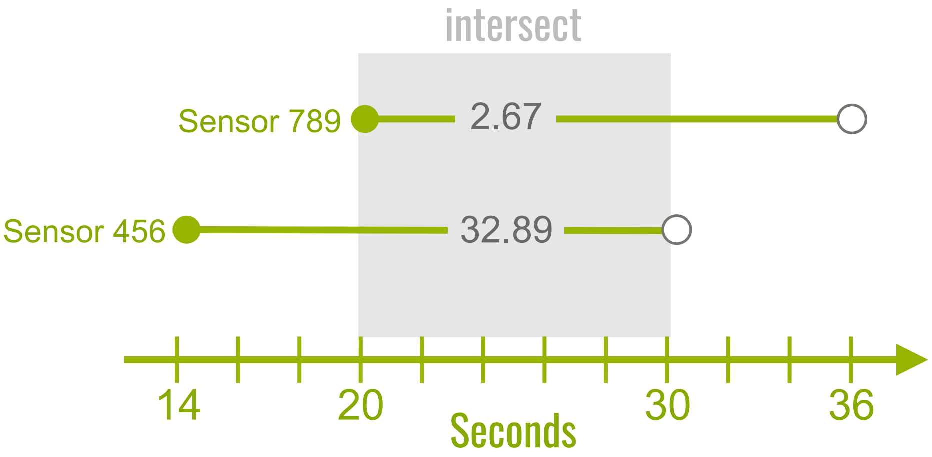
Also, the end time was adjusted.

The above normalization table shows 104 rows were collapsed and later 44 rows also had identical values.

SOLUTION
TEMPORAL INTERSECTION

Temporal intersection reveals when critical events occur in time.

For example, the figure shows the intersection of two sensors #789 and #456. The resulting intersection period is queried by the data scientist who is looking for pre-determined intervals, correlations, or specific incidents at a known point in time.



Flight ID	Sensor ID	Valid Period	Value
123456	789	(2016-11-09 00:00:20.000, 2016-11-09 00:00:36.000)	2.67
123456	456	(2016-11-09 00:00:14.000, 2016-11-09 00:00:30.000)	32.89
		- - - more similar rows - - -	

The following SELECT statement finds all the exact intersect periods:

```
SELECT FLIGHT_ID, SENSOR_ID,
  FROM A INNER JOIN B ON A.PERIOD
OVERLAPS B.PERIOD

WHERE A.SENSOR = 789 A.VALUE > 2.6 and
B.SENSOR = 456 AND B.VALUE > 32;
```

A critical condition could be when both readings exceed certain values.

The secret is simple...

Using a SELECT statement that **uses a temporal join to test whether two** periods intersect with the OVERLAP function. The SQL optimizer can then construct a query execution that efficiently generates the result.

SOLUTION
BOEING DATA PIPELINE

This is the data pipeline used at Boeing to process FDR data

The binary-encoded FDR data is loaded onto the Hortonworks® Hadoop platform. Java code is applied in Hadoop for both decoding the binary format and performing temporal normalization to build time periods into Apache® Hive™ datasets.

Normalization logic is built into the Java load processing to generate periods as a pair of time-value strings.

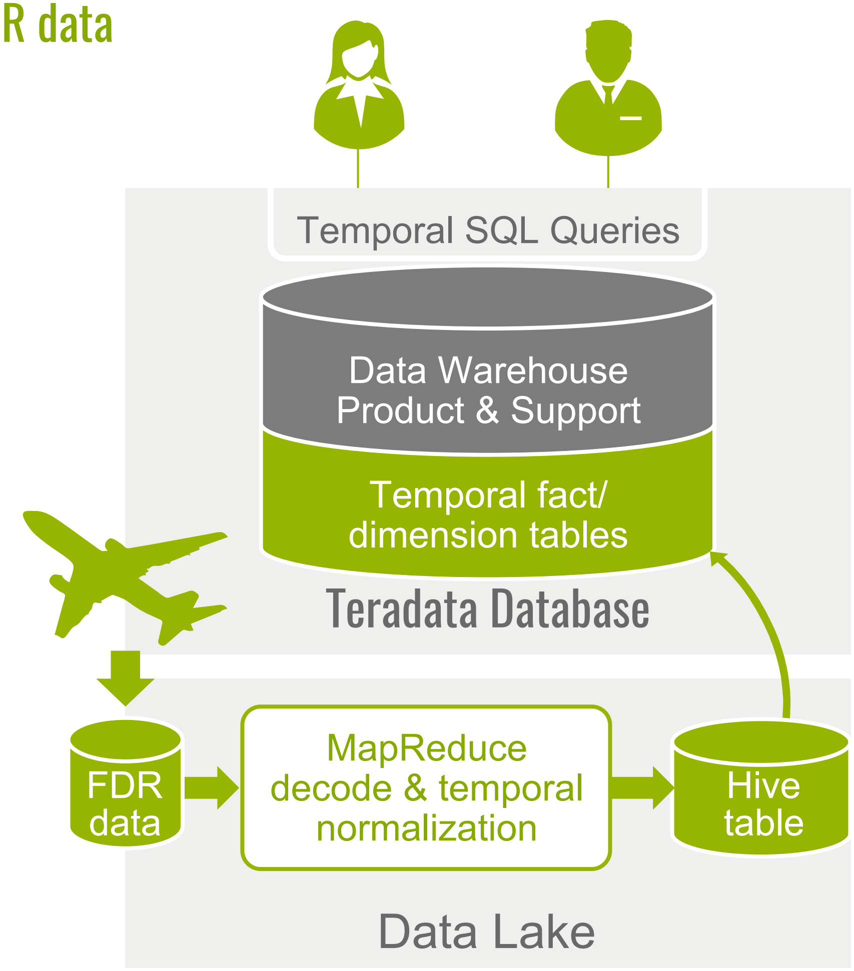
Teradata QueryGrid™ does a parallel transfer from Hive to further transform the data into dimensional tables within the Teradata Database.

The string pairs are converted into proper PERIOD datatypes with:

```
  INSERT SELECT PERIOD
        (TS1-string, TS2-string);
```

By structuring the temporal data as fact-dimension tables, any combination of flight-sensor-time data can be efficiently selected.

More importantly, this data can now be combined with the entire data warehouse to generate a spectrum of analyses and visualizations across maintenance, scheduling, and other functions.


Temporal SQL Queries
Data Warehouse Product & Support
Temporal fact/ dimension tables
Teradata Database
FDR data
MapReduce decode & temporal normalization
Hive table
Data Lake

Benefits

+ Compression & Efficiency
+ Accuracy & Agility

03

BENEFITS
COMPRESSION & EFFICIENCY

COMPRESSION

Sensor data is compressed without information loss by several orders of magnitude, both in the initial load and in subsequent query executions. For example, the load of an eight-hour flight previously required 15GB of Hive storage, now takes 190MB in the data warehouse – an 80x lossless compression. These high lossless compression rates ease concerns for data storage and movement, which is critical for always-flowing voluminous IoT data.

A typical example of using SQL NORMALIZE in a query step reduced data from 5.8 million rows to 19 thousand rows – that's 292x compression! In contrast, columnar data tables usually achieve a maximum of 8x compression.

80x lossless compression

faster queries by 600x and more

292x compression of rows

EFFICIENCY

Analytics are more efficient because temporal functions are pushed down into the database engines for query optimization and low-level parallel execution. Previously, one query analyzing just one hundred flight hours required 200 hours to execute. With temporal SQL, analytics on 71 flights required only a few minutes. Further, the same analytics on a thousand flights took 17 minutes, implying that processing scales linearly.

In another exercise calculating period intersections, a single join step reduced CPU-seconds by 57% and execution time by 65%, along with similar reductions in rows matched and IO transfer.

With faster queries, business users can run many iterations. This improves accuracy and user satisfaction.

BENEFITS
ACCURACY & AGILITY

ACCURACY

For Boeing, analytics are more accurate because the full resolution of raw sensor data can be utilized, especially for finding period intersections. By avoiding approximating to the nearest whole second, the alignment of those periods is exact. Further, the detection of recording gaps can be clearly indicated by the absence of a valid period. So, a SQL query to list the number and duration of all gaps for Sensor 789 is straightforward.

Accuracy in alignment and gap detection to the millisecond can be critical in an aircraft accident investigation, maintenance planning, and next-generation plane designs.

exact results
at scale!

easy, quick
discoveries

AGILITY

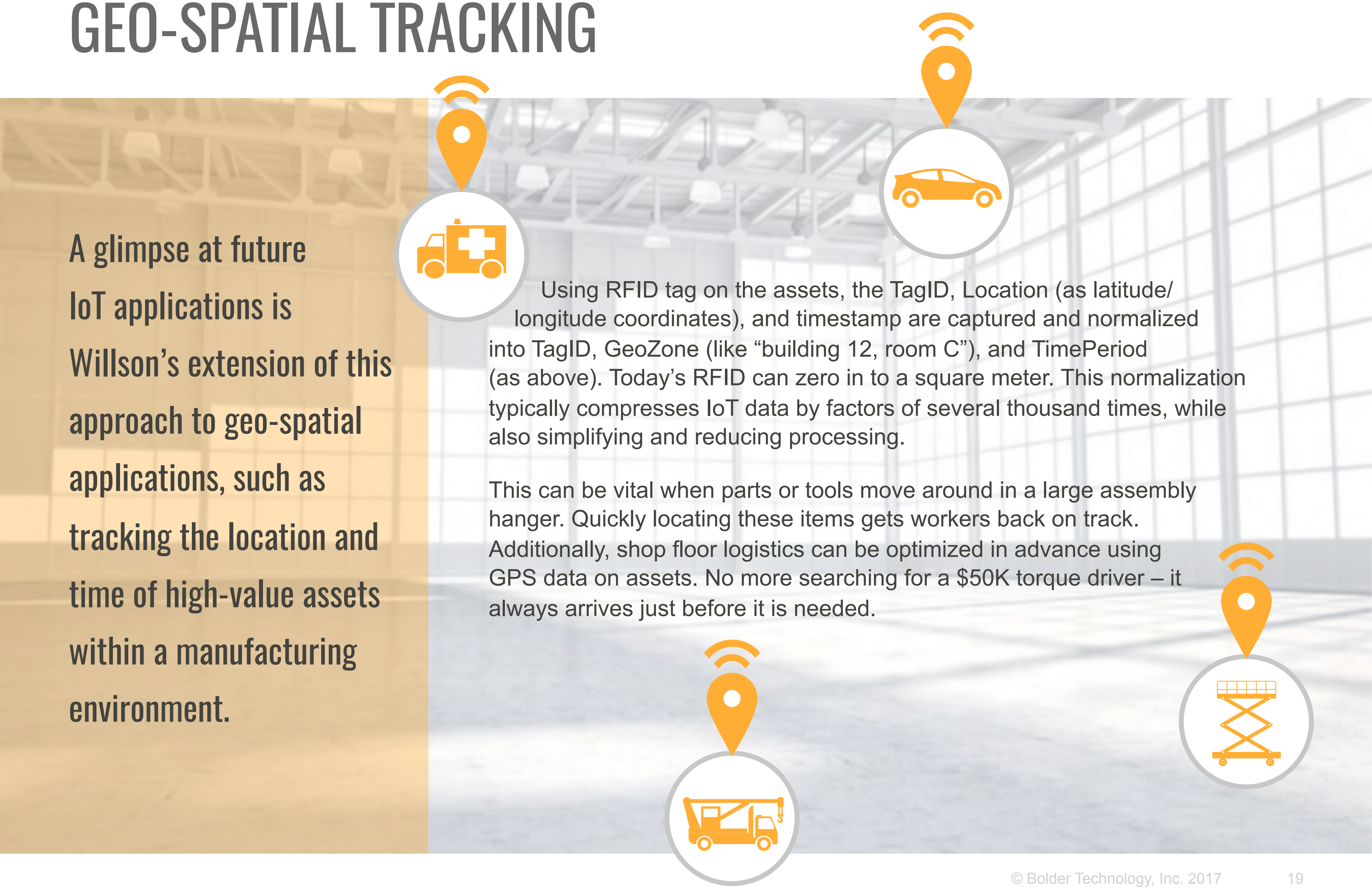
Modern analytics ecosystems become more agile. This is both in its capacity to handle larger IoT problems and in its ability to apply its infrastructure to other IoT applications. By using standardized tools rather than custom code, this IoT application of FDR analytics leverages the existing data warehouse, report generation, and data visualization tools. Plus, data science teams can tap into the IoT data directly using temporal SQL, avoiding delays from custom programming.

Future

- + Geo-Spatial Tracking
- + Systems That Never Break!

04

FUTURE
GEO-SPATIAL TRACKING

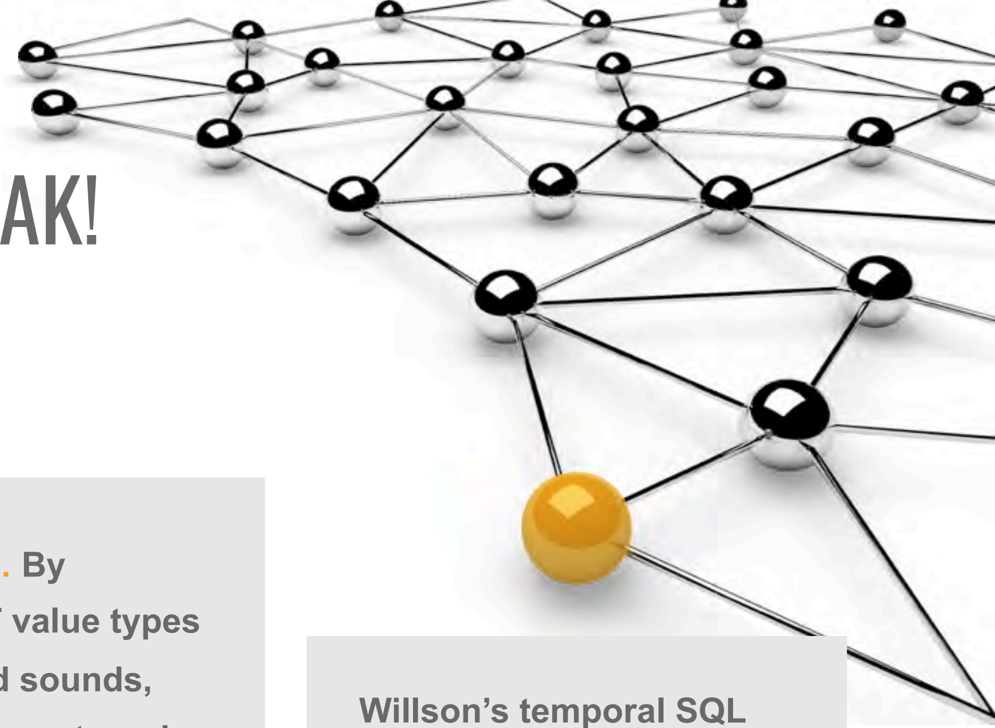


A glimpse at future IoT applications is Willson’s extension of this approach to geo-spatial applications, such as tracking the location and time of high-value assets within a manufacturing environment.

Using RFID tag on the assets, the TagID, Location (as latitude/ longitude coordinates), and timestamp are captured and normalized into TagID, GeoZone (like “building 12, room C”), and TimePeriod (as above). Today’s RFID can zero in to a square meter. This normalization typically compresses IoT data by factors of several thousand times, while also simplifying and reducing processing.

This can be vital when parts or tools move around in a large assembly hanger. Quickly locating these items gets workers back on track. Additionally, shop floor logistics can be optimized in advance using GPS data on assets. No more searching for a $50K torque driver – it always arrives just before it is needed.

FUTURE
SYSTEMS THAT NEVER BREAK!



Willson's innovation of using temporal SQL for IoT analytics is not obvious. Yet, by maintaining raw IoT data in Hadoop and analyzing on the data warehouse, the benefits of storage saving and, especially, query performance are achieved.

In the future… By extending IoT value types to images and sounds, new innovative categories of IoT applications will emerge. The only requirement is a function that can indicate whether two images (or sounds) are equal, thus resulting in effective analytics based on exact time intersects.

Willson's temporal SQL approach to IoT analytics is simplifying the management of complex systems by improving our understanding. By predicting system faults in advance of failure, the dream of complex systems (like aircraft) that never break comes closer to reality.

About the Methodology

The methodology for this study is to listen carefully to innovative companies in data analytics and to report accurately on their perceptions. The intent is to contribute to professional education—to share the experiences and best practices with other IT professionals so that we can mature as an industry, amid escalating business challenges and rapidly evolving technology. In particular**, Ian Willson** of Boeing has graciously shared the technical details of his IoT temporal analytics innovation at his 2016 PARTNERS Conference presentation as well as in his detailed patents.

The author is **Richard Hackathorn** of Bolder Technology. A sincere appreciation is given to Teradata Corporation and, especially, **Dan Graham** for sponsoring this study and for permitting open access to its customer community.

About Bolder Technology

Bolder Technology is an industry analysis firm focusing on Business Intelligence and Business Analytics. Its founder, Richard Hackathorn, has more than thirty years of experience in the Information Technology industry as a well-known industry analyst, technology innovator, and international educator.

About Our Sponsor

Teradata empowers companies to achieve high-impact business outcomes. With a portfolio of business analytics solutions, architecture consulting, and industry-leading big data and analytics technology, Teradata unleashes the potential of great companies.

TERADATA.

EB-7269 > 0517