Database Answers

A Big Data Tutorial for Data Modellers by Example

Barry Williams
barryw@databaseanswers.org

1.What ?

In this Tutorial, we discuss some specific these kinds of Big Data Things to clarify what Big Data is and it is processed.

2.Why ?

I have written this Tutorial because I have a number of requests that will help people who share my background to understand the important topic of Big Data.

3.How ?

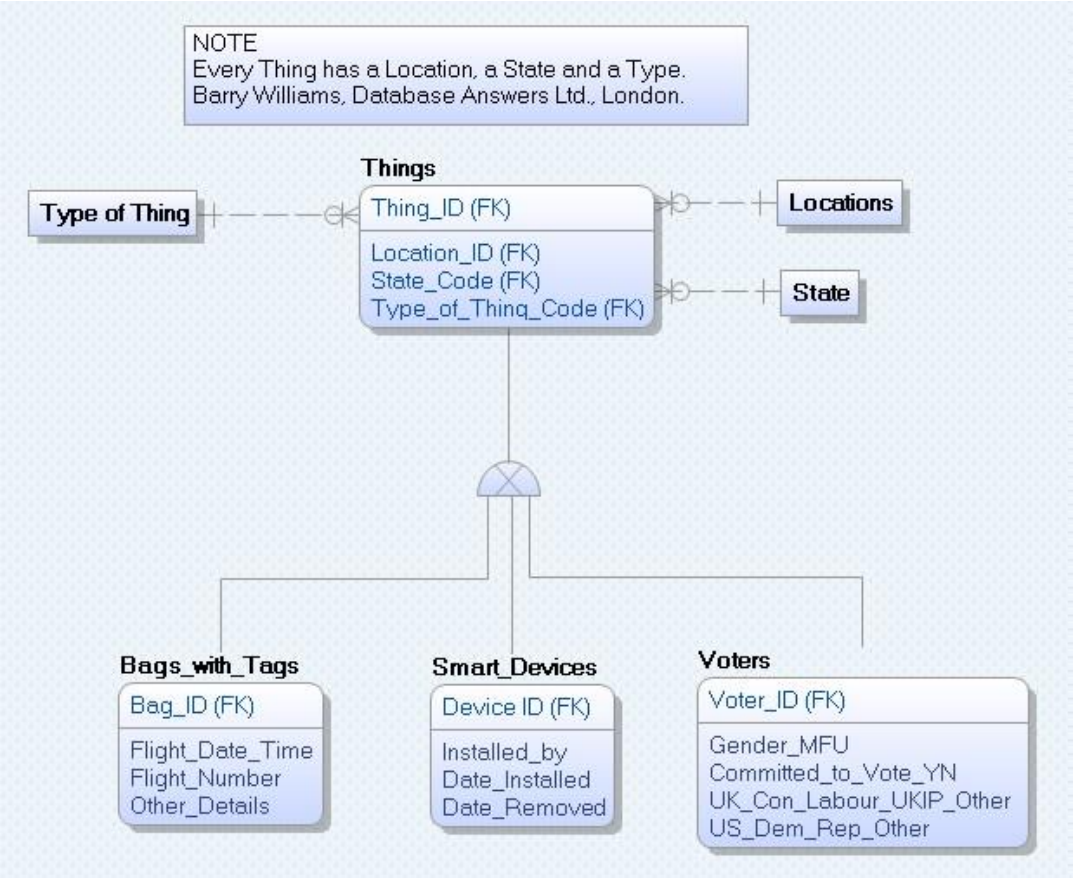
I will be discussing these topics :-

- Bags with Tags
- Smart Devices at Home
- Voters

3.1Data Model

This is an ERD showing Inheritance.

This demonstrates the complexity of the world of Things.

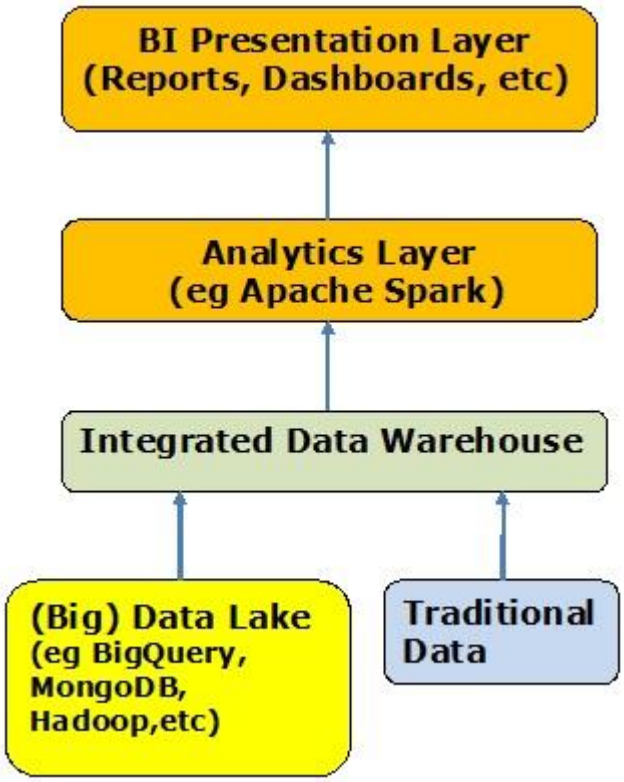


3.2 KPIs

- 'Talking Cheese'
  - Send a Message
- Smart Home Devices
  - % Failed
  - Movement details
  - Status
- Bags with Tags
  - % Damaged
  - % Lost and Found
- Voter's Preferences
  - Candidate's Popularity
  - Voting Intentions

3.3Layered Data Architecture 1 (LDA)

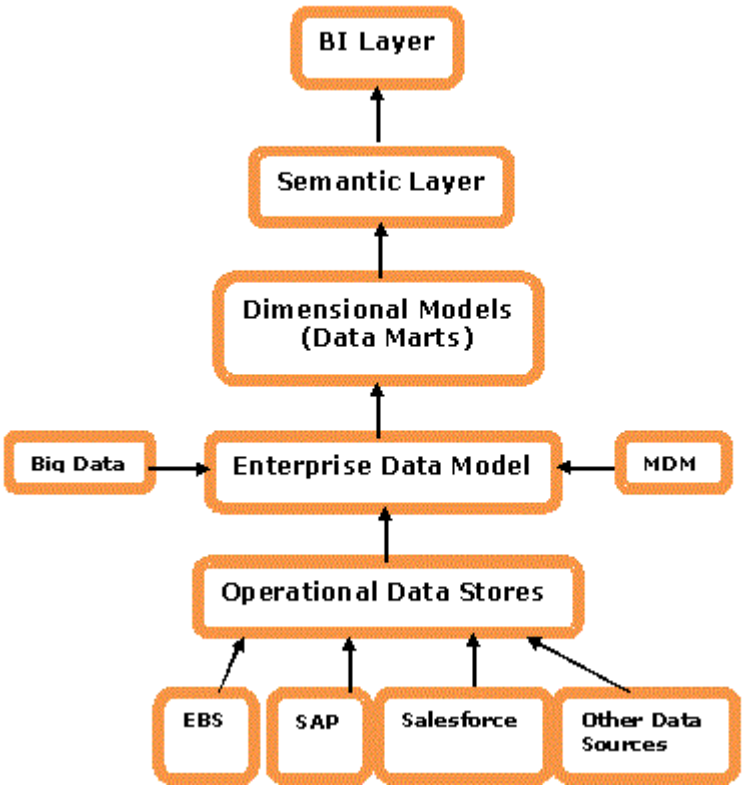
This is a Big Data version of our Reference Data Architecture.



3.5 Reference Data Architecture

This is our Reference Data Architecture on our Database Answers Web Site :-

- http://www.databaseanswers.org/reference_data_architecture.htm



.

3.5 Hadoop on Wikipedia

Wikipedia is (as always) useful reference :-

- https://en.wikipedia.org/wiki/Apache_Hadoop

4. Hadoop and Traditional Technology Stacks

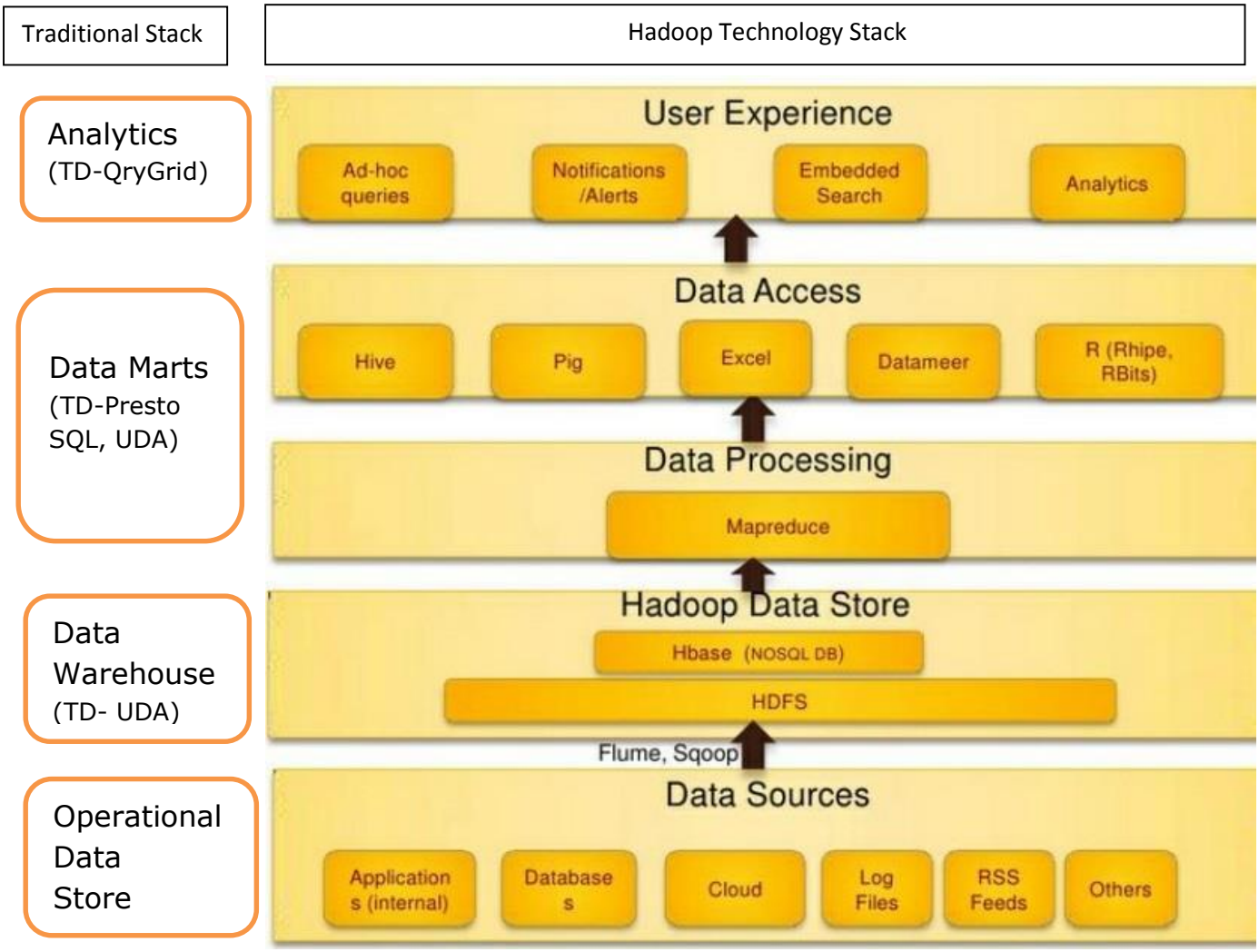
4.1 A Comparison

Here I compare a Traditional Stack against a Hadoop Technology stack which appears on the Hadoop360 Web Site :-

- http://www.hadoop360.com/blog/hadoop-technology-stack

In the left-hand column, I use TD to indicate Teradata Components that are part of the Teradata Unified Data Architecture :-

- http://www.databaseanswers.org/data_models/big_data/Teradata_Unified_Data_Architecture.htm

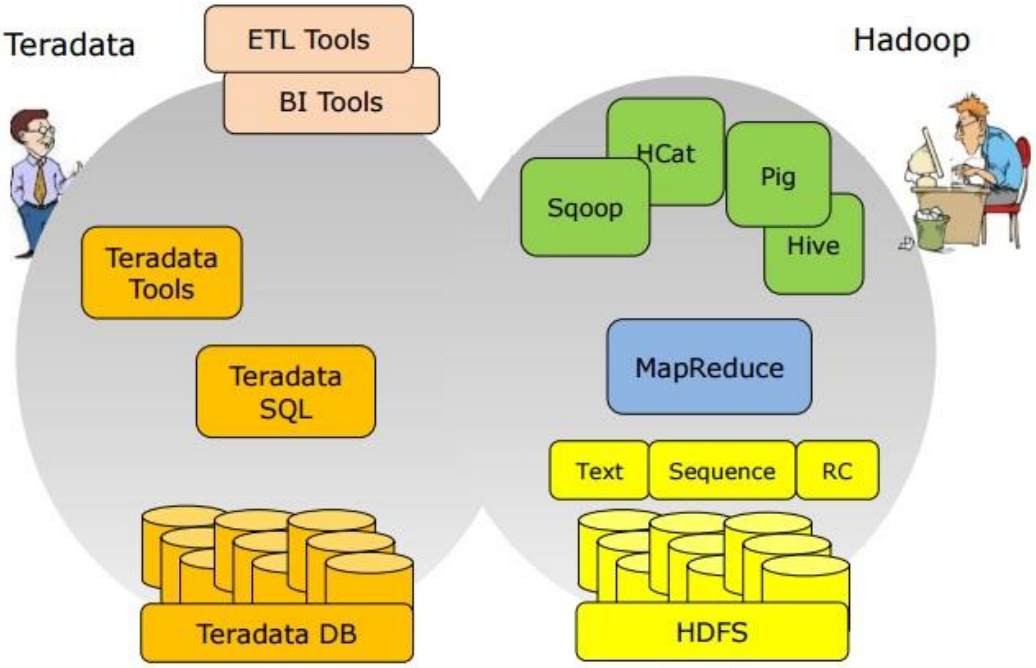


4.1 Common Ground

The material that follows is taken from this PDF document on the Teradata Web Site :-

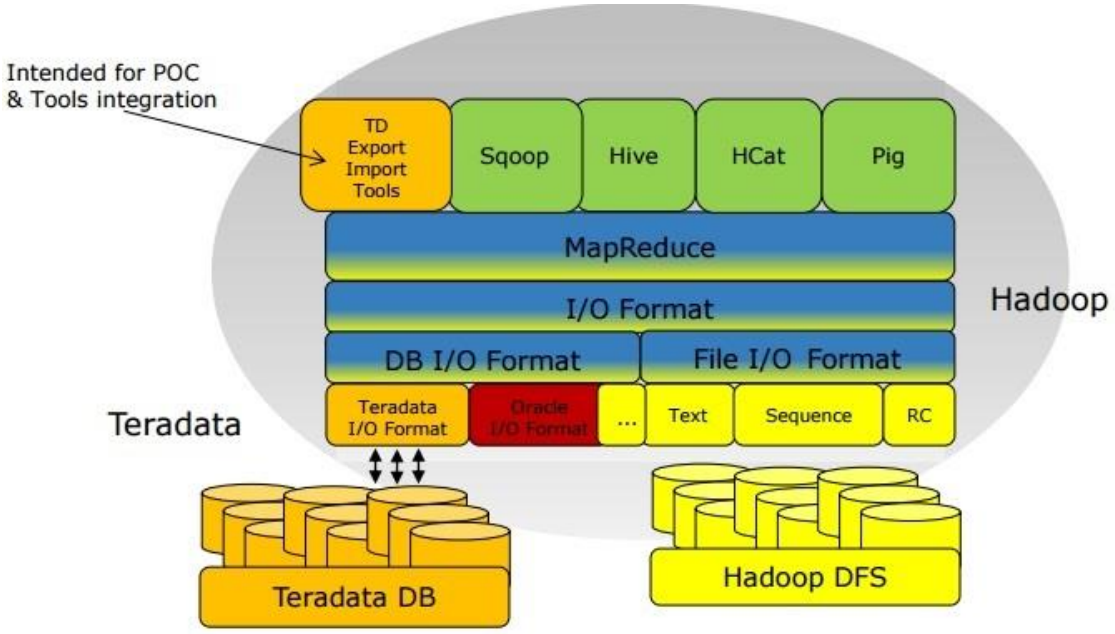
- https://developer.teradata.com/sites/all/files/Teradata%20Connector%20for%20Hadoop%20Tutorial%20v1%200%20final.pdf

This shows the common ground between Hadoop and Teradata :-



4.2 Architecture

MapReduce is a framework for processing parallelizable problems across huge datasets using a large number of computers (nodes).



Teradata v14.0 provides three functions for processing name-value pair data, based on the pioneering work carried out at eBay:
1. NVP: extract name-value pair (with the possibility to specify delimiters other than & and =)
A more complex form of name-value pair data consists of a delimited list of values for a single name, such as "?price=243.00,234.00,239.00,254.00", which might represent four airline ticket prices returned in a search.

The following two functions can be applied:
As one the oldest—founded in 1995—and largest marketplaces on the Web, eBay lives or dies by its ability to understand what its buyers are looking for and its success in linking prospective buyers to sellers. eBay continuously and actively improves the appearance and behavior of its website in order to make the most appropriate offers to prospective buyers and to give the best service to advertisers.
This process is driven by data collected about search terms entered, result sets shown and eventual purchases. The data thus collected is used by business planners to improve page layouts, product placements, and drive higher sales.
eBay needed to get this extensive and rapidly changing data into the hands of its business analysts in a timely and highly operationally reliable manner, and in a form that they could easily manipulate and query. With large-scale, production data warehousing already on a Teradata platform, these requirements strongly suggested that a relational solution would be the optimal approach—if they could process query strings effectively and enable analysis via standard SQL.

2. STRTOK: extract a single item from a delimited list
3. STRTOK_SPLIT_TO_TABLE: transpose or pivot a delimited list into multiple rows of a column

A code example of handling name-value pair data
The following is a more detailed example of the above functions applied to automotive sensor data.
Define table and insert two sample log data lines:

```
create volatile table car_data (log_ts timestamp(6), vin_nbr bigint,
sensor_name varchar(128), payload varchar(4096) character set unicode)
on commit preserve rows;
Insert into car_data('2012-01-01
14:12:31.000000',123,'Fuel','temperature=85&pressure=3&level=0.8');
Insert into car_data('2012-01-01
14:12:31.000000',789,'Cylinders','cyl_list=2,5,6&error=12345');
```

Extract the temperature and pressure for fuel:

```
select log_ts,vin_nbr,nvp(payload,'temperature') as
Temperature,nvp(payload,'pressure') as Pressure from car_data where
sensor_name = 'Fuel';
```

Extract the first cylinder with a malfunction:

```
select log_ts,vin_nbr,strtok(nvp(payload,'cyl_list'),',',1) as
firstBadCyl from car_data where sensor_name = 'Cylinders'
```

log_ts vin_nbr firstBadCyl
1 2012-01-01 14:12:31.000000 789 2

Extract the cylinder ids with a malfunction and return as a set:

```
with bad_cyl_data (vin_nbr,cyl_list)
as (select vin_nbr,nvp(payload,'cyl_list') from car_data where
sensor_name = 'Cylinders')
select * from table
```

(strtok_split_to_table(bad_cyl_data.vin_nbr,bad_cyl_data.cyl_list,',')
returns (vin_nbr int, result_position int, result_item varchar(20)
character set unicode)) **as** split_data
vin_nbr result_position result_item
1 21 1 2
2 21 2 5
3 21 3 6
The advantages of this approach are:
log_ts vin_nbr Temperature Pressure
1 2012-01-01 14:12:31.000000 123 85 3

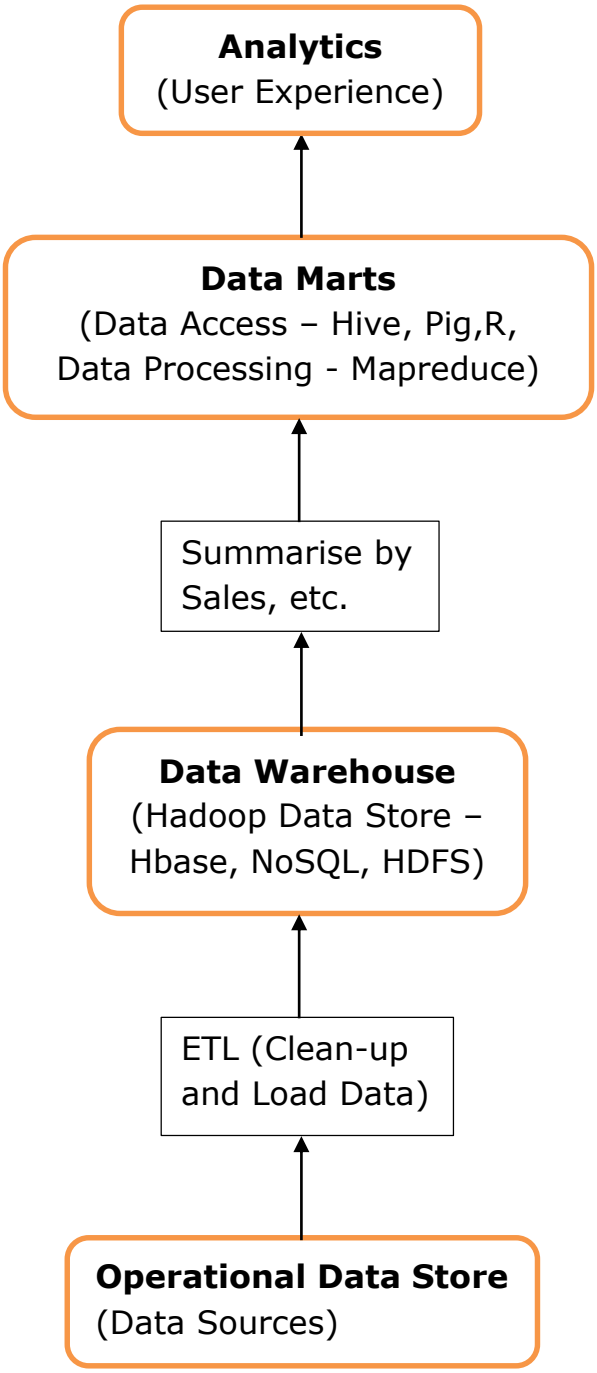
A relational database approach often offers the best combination of features for supporting name-value pair data.

The flexibility inherent in the name-value pair approach is maintained throughout the ETL, data processing and storage. Only at the last moment, at query time, is the string parsed out.

A production-strength relational database is used, benefiting from all the reliability, performance, availability and management features found there. In contrast to a Hadoop solution, which could certainly be coded, the relational approach keeps all data securely controlled and preserved.
No additional storage or processing overhead is added to the relational database to handle name-value pairs.
Block level compression of the data exceeding 80% can be achieved.
Business users gain direct access to the query string data and can manipulate and analyze it as they need. Only common SQL skills are required; as opposed to MapReduce programming skills needed in Hadoop.

5. Data Processing

Here we show the Major Steps in processing data from Source Data into Analytics and the User Experience. We show names of the Hadoop Components in brackets.



Step 1. Load Data into our Operational Data Store

Step 2. Clean-up Data

Step 3. Load Data into our Data Warehouse

5.1 The Requirements

The Requirements can be defined as follows :-

We need to handle 'Big Data' for Bags, Devices in the Home and Voter's Preferences.

- 'Talking Cheese'
    - Send a Message
- Smart Home Devices
    - % Failed
    - Movement details

- Bags with Tags
    - %Damaged
    - % Lost
    - % Lost and Found

- Voter's Preferences
    - Intentions
    - Candidate's Popularity

5.1 Life-Cycle of a Data Set

In this Chapter we present our approach to bridging the gap between 'traditional' Data Management and Data Science.

Basic	'Talking Cheese'
Intermediate	Smart Home Devices
	Bags with Tags
Advanced	Voter's Preferences

6.SQL and the Requirements

The Requirements are as follows.

6.1Bags with Tags

We want to calculate percentages for a specific Type of Device and specific Status of Damaged, Lost, Lost and Found

```
SELECT COUNT(*) / 100 AS Percentage
FROM Things
Where Type_of_Thing = 'Bag with Tag'
```

6.2 Smart Home Devices

```
SELECT COUNT(*) / 100 AS Percentage
FROM Things
Where Type_of_Thing = 'Device'
```

6.3 Voters

7.An SQL Solution

A Solution to implement the KPIs in SQL would look something like this.

Here we can use Teradata's Presto SQL :-

- http://www.teradata.co.uk/PrestoDownload/?LangType=2057&LangSelect=true

Cloudera and Hadoop :-

- http://www.slideshare.net/cloudera/harnessing-the-power-of-apache-hadoop

Hive vs MySQL :-

- http://blog.matthewrathbone.com/2015/12/08/hive-vs-mysql.html

7.1Bags with Tags

We want to calculate percentages for a specific Type of Device and specific Status of Damaged, Lost, Lost and Found

```
SELECT COUNT(*) / 100 AS Percentage
FROM Things
Where Type_of_Thing = 'Baggage'
```

7.4 Smart Home Devices

SELECT COUNT(*) / 100 AS Percentage
FROM Things
Where Type_of_Thing = 'Device'

7.3 Voters

8.A Non-SQL Solution

A Solution to implement the KPIs in SQL would look something like this.

8.1Hadoop Stack

We want to calculate percentages for a specific Type of Device and specific Status of Damaged, Lost, Lost and Found

SELECT COUNT(*) / 100 AS Percentage
FROM Things
Where Type_of_Thing = 'Baggage'

8.2 Personal Data Lakes

- https://securosis.com/blog/nosql-security-understanding-nosql-platforms

Personal Data Lakes

- http://www.eweek.com/enterprise-apps/hadoop-summit-wrangling-big-data-requires-novel-tools-techniques-2.html

"This paper presents Personal Data Lake, a single point storage facility for storing, analyzing and querying personal data. A data lake stores data regardless of their formats and thus provides an intuitive way to store personal data fragments of any type. Metadata management is a central part of the lake architecture. For structured/semi-structured data fragments, metadata may contain information about the schema of the data so that the data can be transformed into queryable data objects when required. For unstructured data, enabling gravity pull means allowing third-party plugins so that the unstructured data can be analyzed and queried.

Personal Data Lake With Data Gravity Pull (PDF Download Available). Available from: https://www.researchgate.net/publication/283053696_Personal_Data_Lake_With_Data_Gravity_Pull [accessed Feb 17, 2016]."

8.3 Smart Home Devices

SELECT COUNT(*) / 100 AS Percentage
FROM Things
Where Type_of_Thing = 'Device'

8.4 Voters

9. Forbes

Interesting blog by Daniel Newman on "Big Data And Buyer's Journey: Measuring The Invisible" :-

- http://www.forbes.com/sites/danielnewman/2015/06/09/big-data-and-the-buyers-journey-measuring-the-invisible/#298e18723b52

and another one on "What should Digital Transformation mean to your Business" :-

- http://www.forbes.com/sites/danielnewman/2015/11/11/what-should-digital-transformation-mean-to-your-business/#99cb2f246301

10.Supplier-specific Solutions

10.1 IBM

10.2 Microsoft

Microsoft has published a paper entitled "The Microsoft Modern Data Warehouse" in which they say (on page 15) "According to Gartner, " Big Data is high volume, high velocity, and/or high variety information assets that require new forms of processing to enable enhanced decision making, insight discovery, and process optimization"

Figure 9: Using Microsoft StreamInsight to process large event streams

10.3 Oracle

Microsoft has published a paper entitled "The Microsoft Modern Data Warehouse" in which they say

10.4 Teradata

Microsoft has published a paper entitled "The Microsoft Modern Data Warehouse" in which they say

10.5 Kaggle Datasets

Looks interesting :-

- https://www.kaggle.com/datasets

11. For Future Requirements

11.1 As required.

Wikipedia has a definition of Data Lakes :-

Microsoft has published a paper entitled “The Microsoft Modern Data Warehouse” in which they say

12. Links

12.1 Apache Hadoop Stack

http://www.slideshare.net/rohitkulky/scaling-up-with-hadoop-and-banyan-itrix-feb-2015-public-copy

12.2 Hadoop in Wikipedia

Microsoft has published a paper entitled “The Microsoft Modern Data Warehouse” in which they say

- https://en.wikipedia.org/wiki/Apache_Hadoop

12.3 Wikipedia on Data Lakes

Wikipedia has a definition of Data Lakes :-

- https://en.wikipedia.org/wiki/Data_lake

The term was coined by James Dixon, Pentaho chief technology officer.[2] Dixon used the term initially to contrast with "data mart", which is a smaller repository of interesting attributes extracted from the raw data. He wrote: "If you think of a datamart as a store of bottled water – cleansed and packaged and structured for easy consumption – the data lake is a large body of water in a more natural state. The contents of the data lake stream in from a source to fill the lake, and various users of the lake can come to examine, dive in, or take samples." [3] Dixon argued that data marts have several inherent problems, and that data lakes are the optimal solution.

Dixon identified 2 shortcomings of data marts: "Only a subset of the attributes are examined, so only pre-determined questions can be answered." and "The data is aggregated so visibility into the lowest levels is lost."
Data Lakes are the Big Data replacement for Data Marts.
"Enterprises across industries are starting to extract and place data for analytics into a single, Hadoop based repository."
PricewaterhouseCoopers also claim that cost is a major reason that organizations adopt data lakes they state that:
Some vendors that advocate the use of Hadoop claim that the cost per terabyte for data warehousing can be as much as $250,000, versus $2,500 per terabyte (or even less than $1,000 per terabyte) for a Hadoop cluster.[4]

Currently the only viable example of a data lake is Apache Hadoop.

Appendix A. Mapping Traditional and IoT Data Sets

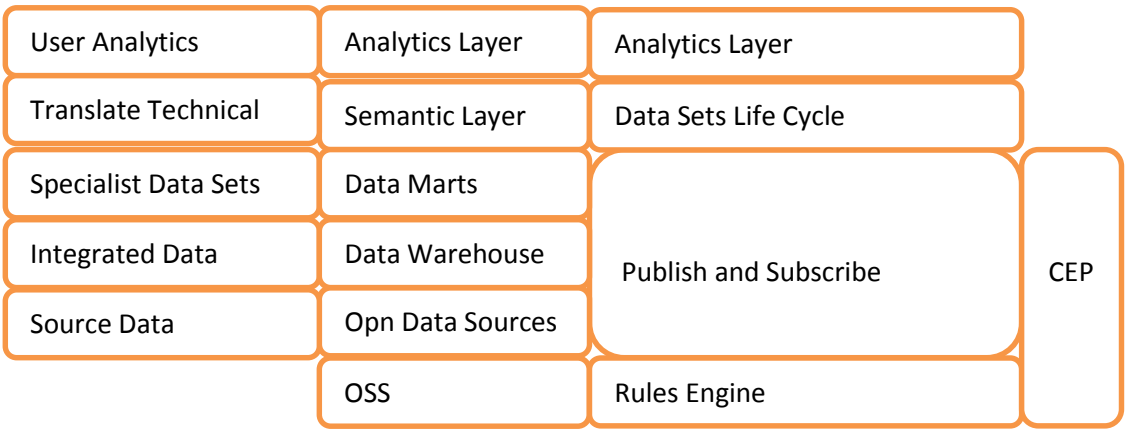
Here we present our mapping of our Traditional Best Practice Data Sets

This is not straightforward because the logic involved does not match in a one-to-one manner.

For example, we can see that Publish and Subscribe is used in the Data Marts, the Data Warehouse and the Operational Data Sources (ODS).

The CEP applies to the Publish and Subscribe and to the Rules Engine.

.



Appendix B. Hadoop

7.1Microsft Modern Data Warehouse

Hadoop is open-source software that is widely used to process Big Data.

“The Microsoft Modern Data Warehouse integrates Hadoop to provide the ability to seamlessly manage relational and non-relational data from a shared query model, infrastructure and ecosystem.”

Appendix C. Extract from Teradata's Paper

This is an extract from pages 9 and 10 in this paper :-

- https://site.teradata.com/Microsite/Unlocking-Machine-Generated-Data/Landing/.ashx

Handling name-value pair data in the Teradata database at eBay

By 2008, eBay was facing a challenge posed by the growth rate in web log records. The value of such data was already proven. IT could and did deliver it using an ETL-based approach.
But, now, business users wanted to analyze and play with that data themselves in tools they already knew, rather than have IT create batch analytic jobs or reports on their behalf.
IT needed a way to make the data available for more people, to scale to petabytes and to improve flexibility to changing data. A number of alternative approaches were investigated, but none offered the overwhelming benefits that could be achieved if the name-value pair data from query strings could be processed effectively in a relational database and offered to end users via standard SQL.
Two typical relational approaches were initially considered.
The first was to extract each unique name-value pair into a specific column based on the name in the pair.
The resulting table would, however, be extremely sparse and some tens of thousands of columns wide. As new name-value pairs were created, the table and load process would have to be altered on an ongoing basis. The ongoing maintenance and storage requirements led to the rejection of this tactic. The second approach considered was based on two tables, table 1 with the log message key and any fixed, well-defined attributes; and table 2 being a long table with a row for each name-value pair, with one column containing names and another the values, all linked via a foreign key relationship to table 1. This approach was rejected due to ETL workload, query complexity and database performance concerns.

The final solution is much simpler, has better storage and

performance characteristics and maintains the flexibility of the name-value pair method.

The query string is simply stored as-is in a Varchar(4096) field and specialized SQL functions have been included in the database (generally available in Teradada v14.0) to parse out specific name-value pairs at query time.

Thus, for example, applying the SQL function "Select nvp(query_string,'vo') from logtable;" to a query string containing "?h=c8&g=739d69c81&c=1&vo=117933&rnm=0…" returns the value 117933. In addition to the Varchar string, the table contains standard key and timestamp columns, as well as other fixed fields from the web log. To further improve usability and performance at query time, common name-value pairs that seldom if ever change can be extracted to their own columns at load time and used directly by business users.

Teradata v14.0 provides three functions for processing name-value pair data, based on the pioneering work carried out at eBay:

1. NVP: extract name-value pair (with the possibility to specify delimiters other than & and =)

A more complex form of name-value pair data consists of a delimited list of values for a single name,

such as "?price=243.00,234.00,239.00,254.00", which might represent four airline ticket

prices returned in a search.

The following two functions can be applied:

As one the oldest—founded in 1995—and largest marketplaces on the Web, eBay lives or dies by its ability to understand what its buyers are looking for and its success in linking prospective buyers to sellers. eBay continuously and actively improves the appearance and behavior of its website in order to make the most appropriate offers to prospective buyers and to give the best service to advertisers. This process is driven by data collected about search terms entered, result sets shown and eventual purchases. The data thus collected is used by business planners to improve page layouts, product placements, and drive higher sales.
eBay needed to get this extensive and rapidly changing data into the hands of its business analysts in a timely and highly operationally reliable manner, and in a form that they could easily manipulate and query. With large-scale, production data warehousing already on a Teradata platform, these requirements strongly suggested that a relational solution would be the optimal approach—if they could process query strings effectively and enable analysis via standard SQL.

2. STRTOK: extract a single item from a delimited list

3. STRTOK_SPLIT_TO_TABLE: transpose or pivot a delimited list into multiple rows of a column

A code example of handling name-value pair data

The following is a more detailed example of the above functions applied to automotive sensor data.

Define table and insert two sample log data lines:

create volatile table car_data (log_ts timestamp(6), vin_nbr bigint,

sensor_name varchar(128), payload varchar(4096) character set unicode)

on commit preserve rows**;**

Insert into car_data('2012-01-01

14:12:31.000000',123,'Fuel','temperature=85&pressure=3&level=0.8')**;**

Insert into car_data('2012-01-01

14:12:31.000000',789,'Cylinders','cyl_list=2,5,6&error=12345')**;**

Extract the temperature and pressure for fuel:

select log_ts,vin_nbr,nvp(payload,'temperature') **as**

Temperature,nvp(payload,'pressure') **as** Pressure **from** car_data **where**

sensor_name = 'Fuel'**;**

Extract the first cylinder with a malfunction:

select log_ts,vin_nbr,strtok(nvp(payload,'cyl_list'),',',1) **as**

firstBadCyl **from** car_data **where** sensor_name = 'Cylinders'

log_ts vin_nbr firstBadCyl

1 2012-01-01 14:12:31.000000 789 2

Extract the cylinder ids with a malfunction and return as a set:

with bad_cyl_data (vin_nbr,cyl_list)

as (**select** vin_nbr,nvp(payload,'cyl_list') **from** car_data **where**

sensor_name = 'Cylinders')

select * **from table**

(strtok_split_to_table(bad_cyl_data.vin_nbr,bad_cyl_data.cyl_list,',')

returns (vin_nbr int, result_position int, result_item varchar(20)

character set unicode)) **as** split_data

vin_nbr result_position result_item

1 21 1 2

2 21 2 5

3 21 3 6

There are advantages to this approach.