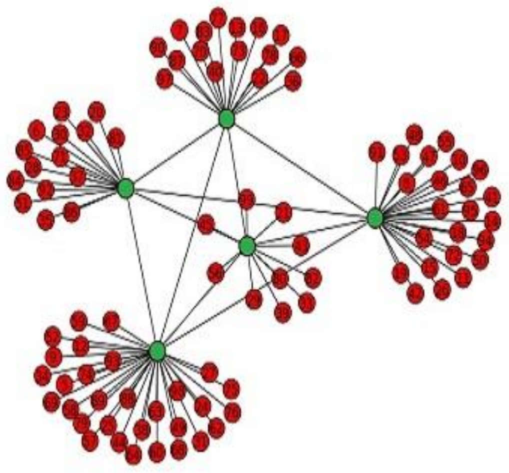
What Is FDN Distributed Analytics?

The future of data management requires a new approach to data sharing based on distributed and decentralized information. *Federated Data Networks* from the *UbiVault Fabric* provides the framework for analytics across many industries where big data is pervasive. The fabric is designed for peer-to-peer data exchange.



What Is FDN Distributed Analytics?

Introduction

Analytics is the discovery of useful information from a data source. Traditional approaches require a centralized data source as a target. The source may be in various formats depending on the method used to source and access data. This may be a flat file, database, or a format compatible with tools such as Excel, CSV, JSON, PDF.

A distributed data source involves the collection, formatting and analysis of data gathered from multiple locations. These can be IOT devices such as sensors, data generation from SMART energy components, institutional data collectors such as hospitals, clinics and healthcare data repositories, military and government data networks and so on.

Challenges

A key challenge facing analytics is the custodian and security issue of who collects and provides data. The data owner may be prevented or is reluctant to allow its data to be passed to a third party for analytics. This issue often occurs when data crosses geographical boundaries to be shared by entities for a common goal. Examples are, terrorist tracking and detection, medical data for a defined purpose, financial transactions, or data that is used to construct profiles for building an understanding of a person or entity.

Principle of Federated Data Networks

To address security and custodian issues, **FDN** follows the principle of edge analytics. The assumption is that data at the atomic level is never passed to a centralized analytics process where a discovery tool scans atomic level data to create information for analysis. How is this done?

First, we define the term atomic as the "lowest level into which data can be divided". For each data record, there is some form of identifier that provides a unique signature. This identifier may be assigned, computed or translated from the data elements in the data record.

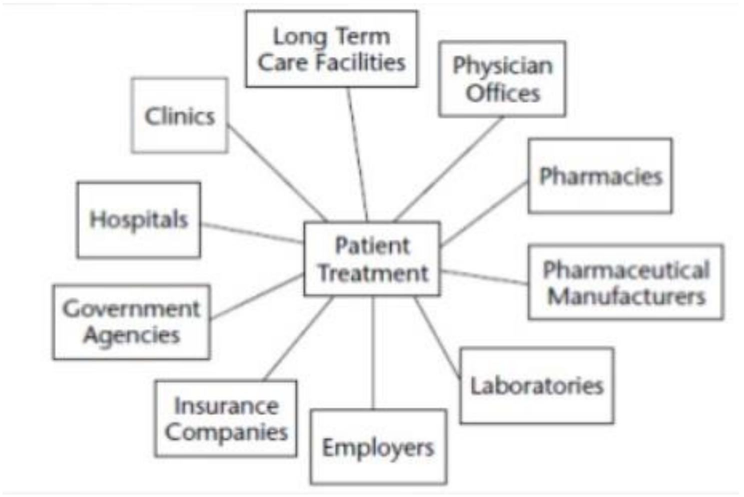
- ✓ An assigned identifier example is a customer id, or social security number.
- ✓ A computed identifier example is a hash of data elements that provide a unique value.
- ✓ A translated identifier is one which uses lookup tables associated with data elements in the data record to construct a unique signature.

The FDN allows a centralized or decentralized analytics process but does not store the atomic level data in a centralized resource. Rather, the analytics process has access to this data, managed and made available at the source or edge, but does not keep the source data or extract to generate a desired information model. To do this requires a basic understanding of structured query language (SQL).

Queries are a form of calculus exploring the relationships among data attributes in a source.

What Is FDN Distributed Analytics?

In simple terms, a data source is comprised of data elements (aka attributes) A data attribute has a known position in the source, a data type, data length and how it is treated for analysis (to be counted or summed.) In relational queries this defines the attributes can be treated as dimensions or as fact tables. A common approach to this data model is the use of a star/snowflake schema. At the center, the atomic level record contains attributes that reflect one or corresponding fact tables. Below is an example for healthcare life-cycle analytics.



The second principle of the FDN is aggregation. This defines that the analytic information is no longer at the atomic level but reflects a count or sum of any query result. In a typical query, the analyst requests a count of atomic records filtered across one or more attributes using SQL syntax. Using the above schema for example,

"How many PATIENTS covered by an INSURER went to one or more LABORATORIES or CLINICS referred to by a particular DOCTOR OFFICE for a PHARMACEUTICAL prescription."

The answer is provided as a COUNT. Each new query will result in another count or summed metric. There is nothing unique to this approach. One performs the query to data stored at the edge, either from a central query tool or executing the query tool at the edge.

Query Object

The query object (QO) is a multidimensional data set defined as all possible combinations of data attribute values having a corresponding non-null count. In mathematics it is referred to as a "tuple". The result is a count of how many atomic records share this combination of data attribute values.

A first order tuple shows how many records have a single attribute value or set of values. A second order tuple is the interest of the first attribute with another attribute. If the QO has N data attributes, then the N-th order tuple describes the intersection of all N attributes and chosen set of attribute values. This defines a relational query as supported with SQL.

What Is FDN Distributed Analytics?

Tuple 1	Tuple 2	Tuple 3
Records with select values for attribute 1	Records with select values for attribute 1 and 2	Records with select values for attribute 1,2 and 3
Records with **Gender**=Female	Records with **Gender**=Female and **age** 18-35	Records with **Gender**=Female and **age**= 18-35 and **Income**= $20K - $50K

Sparse Matrix Problem

There are many possible tuples. Here is the product of the set of enumerated values among 2 attributes$T_2 = (a_1 \cap a_2)$. For an N attribute space the product is defined as the probability of a count of records satisfying $T_n = (a_1 \cap a_2 \cap a_3 \cap a_{4,,,} \cap a_N)$.

The relational model (rows and columns) must reserve space for null values in a data attribute. The QO model stores only the existing tuples of records that have non-null counts. If a tuple set does not occur in the source data, it is not recorded in the QO. This solves the sparse matrix issue. The QO can be queried using SQL just as if the query was against a relational file.

A second distinction is that the QO is accessed through a single read, unlike relational models that require scanning the entire file with or without indexing (another overhead.) Why this is important to analysts leads to the third advantage: query complexity.

Query Complexity

One of the constraints in analytics is the process of iterative discovery, asking questions that explore and target a group of records.

A SQL query has 3 overheads: (1) which attributes are selected for the query, (2) the sets of values for the attributes, and (3) the filtering rules applied to the query. In data mining, analysts drill down in arbitrary attribute order to construct a query. This can lead to the well-known problem of “runaway queries” that become expensive with large data sets. But

A query to a QO has no such impediment. Every query, no matter how simple or complex, involves a single “get” of a tuple. If it exists, i.e. has a non-null count, the cost to extract is independent of the query’s complexity. If the tuple is by design null (no records had that combination of attribute values), the time to get a count is the same.

QO Benefits

The advantage when big data is a source is that the size of a QO grows in log·log terms, unlike a relational data store which grows much faster in file length and processing overhead.

When first applied to call detail records for telecom carriers, the QO advantages were readily visible. With billions of records created daily, loading the information into a relational database sometimes exceeded a 24-hour period. The processing costs and hardware/software investment using servers and software licenses from IBM, Oracle, Sybase, Informix were enormous compared with equipment needed to create and maintain QOs.

Typical load times to build the RF was 10 times faster at 1/50th the cost. Building a QO representing several billions of records could be done in an hour using dual 64bit RISC processors. Once the QO was available, analysis consumed no further IT resources, as the data object was portable and could run from any laptop or desktop.

More important was that analysts could use the same SQL syntax to interrogate the QO, which could also be joined to other QOs with shared data attributes. There was no need for training to use this alternative to relational queries.

FDN QOs

The data sources, as mentioned earlier, stay at their source under custody of the data provider and are never centralized in a single environment. This data at the "edge" lends itself to the idea of performing analytics in a "cloud" using QOs as the data source. As the QOs are in a sense "views" of the data, any query answer that is non-null, is extracted with a single read.

This takes away the discovery paradox where M queries have to be made before the right result occurs at M+1. In terms of cost, there is no penalty from asking stupid questions until you end up with the right one. Note that QOs can exist at the edge or in the cloud depending on data architecture. The build of a QO can be also at the edge or centralized.

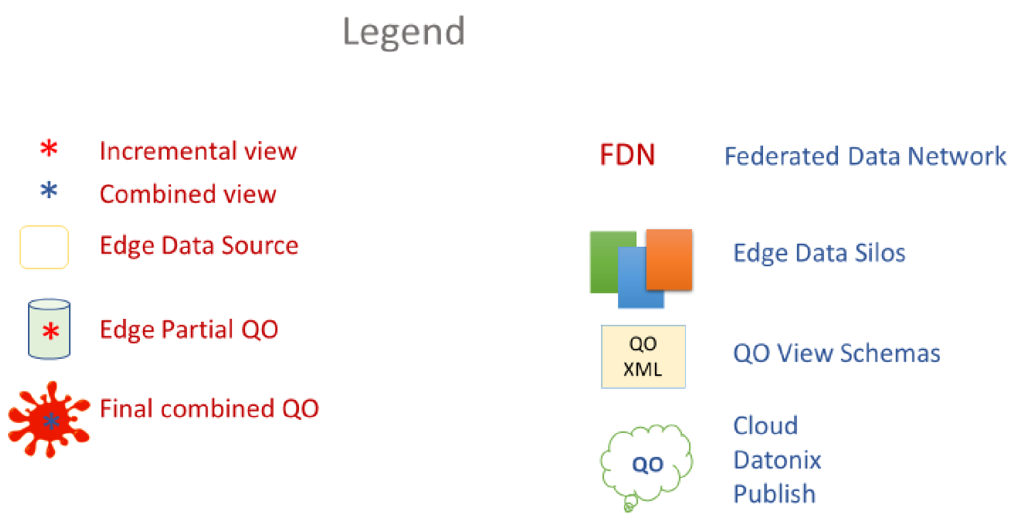
Federation in Practice

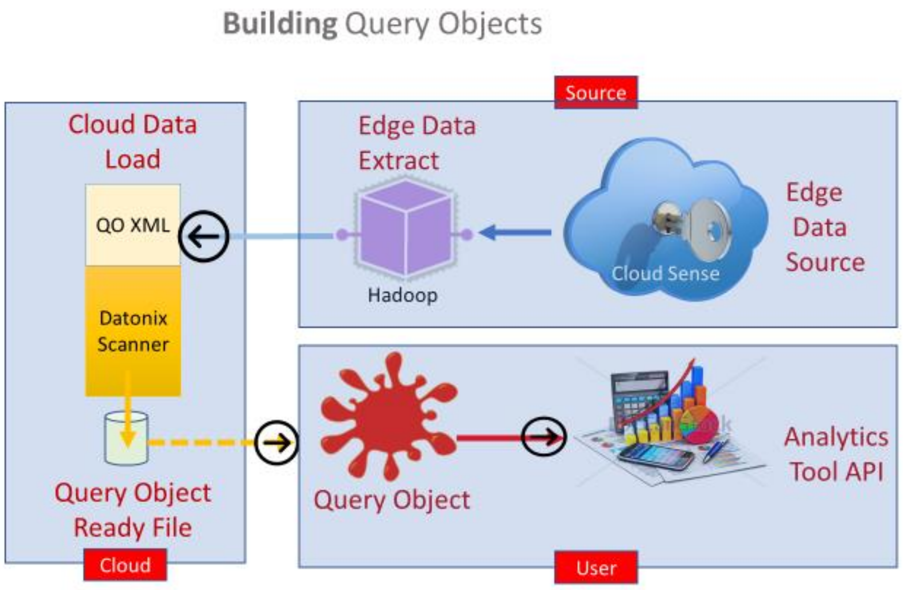
A healthcare example sheds some light on the principle of "federation". Suppose one has two healthcare data providers, each with hundreds of millions of patient or insurance records. Using the process described, each data provider has the ability to create many QOs depending on the attributes of interest. Lets assume the QOs are stored in the cloud for analysis.

If two or more QOs share one or more data attributes (normalized)in its structure, the QOs can be "joined" into a single virtual QO. When the QOs come from different data providers, the analyst has available separate or combined counts and metrics. While there is some slight overhead to the query response, it is nominal depending on how many joins are performed.

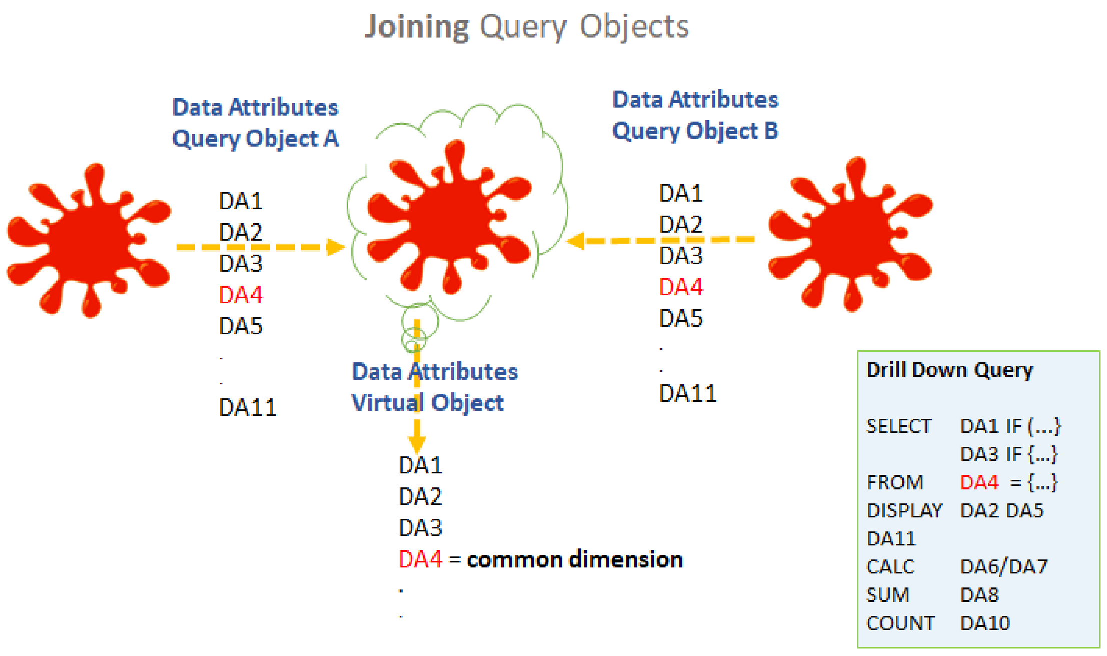
What Is FDN Distributed Analytics?

The Images below define the symbols, and a schematic of building and joining QOs .





What Is FDN Distributed Analytics?



FDN Design

The FDN design keeps the atomic level records at their source. A QO is built by accessing the edge resource and building there an interim (RF) object that is normalized equivalent of the source.

The RF is then used to generate a QO. By definition, data now is no longer at the source record level but is aggregated in the form of tuple sets. In effect, we are storing all possible non-null unions (∩.∩.∩) for the attributes. In simple terms, we store all possible query answers that are non-null in the QO.

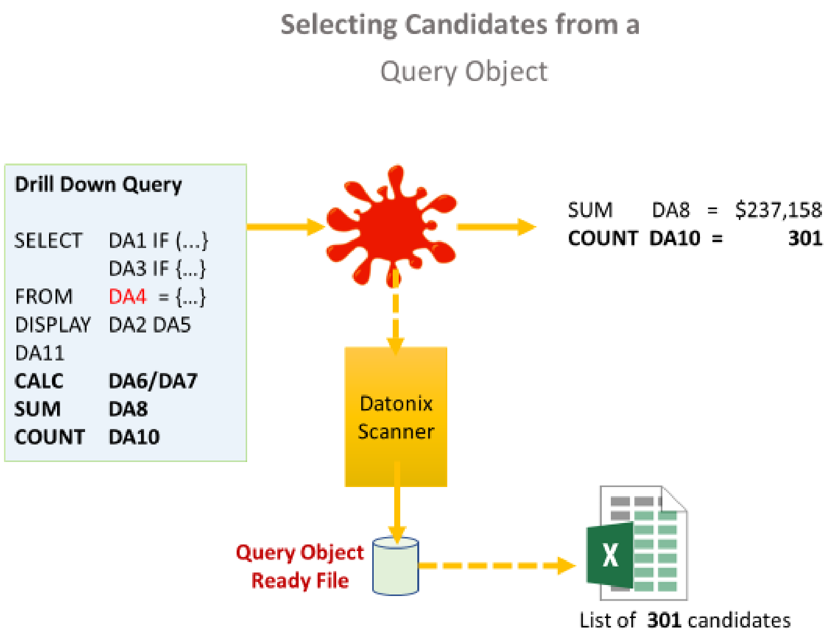
Extracting Records from one or more QOs

While metrics across data attributes are useful for analysis, how can we extract the individual records reflected by a metric or count representing a target group based on the query?

This post analysis step selects the atomic level records from the RF in a single pass. Each qualifying record is passed to a target file that can be used by an application in a normal fashion. When RF is larger, multiple instances of the extraction can be launched in parallel. In a centralized cloud, the extractors query the RF file at the edge and results are stored there.

What Is FDN Distributed Analytics?

The image below describes the data flow.



Summary

FDNs are an excellent way of sharing data without centralization. The use of Query Objects supports analytics while preserving the privacy and control of source data by the owner. The objects support SQL queries and interconnect with third party tools such as spreadsheets, statistical libraries, data visualization and web services.

Author: Andre Szykier Chairman UbiVault