# Franz Inc. to Present at The Global Graph Summit and Data Day Texas

Dr. Jans Aasman, CEO, Franz Inc., will be presenting, "Creating Explainable AI with Rules" at the Global Graph Summit, a part of Data Day Texas. The abstract for Dr. Aasman's presentation:


> *"There's a fascinating dichotomy in artificial intelligence between statistics and rules, machine learning and expert systems. Newcomers to artificial intelligence (AI) regard machine learning as innately superior to brittle rules-based systems, while the history of this field reveals both rules and probabilistic learning are integral components of AI. This fact is perhaps nowhere truer than in establishing* explainable AI*, which is central to the long-term business value of AI front-office use cases."*
>
> *"The fundamental necessity for explainable AI spans regulatory compliance, fairness, transparency, ethics and lack of bias – although this is not a complete list. For example, the effectiveness of counteracting financial crimes and increasing revenues from advanced machine learning predictions in financial services could be greatly enhanced by deploying more accurate deep learning models. But all of this would be arduous to explain to regulators. Translating those results into explainable rules is the basis for more widespread AI deployments producing a more*

*meaningful impact on society.”*

The Global Graph Summit is an independently organized vendor-neutral conference,  bringing leaders from every corner of the graph and linked-data community for sessions, workshops, and its well-known before and after parties.  Originally launched in January 2011 as one of the first NoSQL / Big Data conferences, Data Day Texas each year highlights the latest tools, techniques, and projects in the data space, bringing speakers and attendees from around the world to enjoy the hospitality that is uniquely Austin. Since its inception, Data Day Texas has continually been the largest independent data-centric event held within 1000 miles of Texas.

---

# Big Data 50 – Companies Driving Innovation in 2019

Franz Inc. is proud to announce that it has been named to Database Trends and Application (DBTA) – Big Data 50, Companies Driving Innovation in 2019


Today, more than ever, businesses rely on data to deliver a competitive edge. The urgency to compete on analytics has spread across industries, fueled by the need for greater efficiency, agility and innovation,”

remarked Thomas Hogan, Group Publisher at Database Trends and Applications. “This list seeks to highlight those companies that are really driving innovation and serve as a guide to businesses navigating the rapidly changing big data landscape.”

A new generation of tools is making it possible to leverage the wealth of data flowing into organizations from a previously unimaginable range of data sources. Machine learning, AI, Spark, and object storage are just some of the next-generation approaches gaining traction, according to recent surveys conducted by Unisphere Research, a division of Information Today, Inc.

But, it is also increasingly clear that there is no single way to approach data-driven innovation today. Open source-based technologies have gained strong adoption in organizations alongside proprietary offerings, data lakes are increasingly being implemented but data warehouses continue in widespread use, and hybrid deployments spanning cloud and on-premise are commonly accepted.

Organizations are seeking to use data-driven innovation for better reporting and analytics, real-time decision making, enhanced customer experience and personalization, and reduced costs. But with data coming in from more places than ever, being stored in more systems, and accessed by more users for a wider array of use cases, there is greater recognition that security and governance must be addressed intelligently.

Evaluating new and disruptive technologies, and then identifying how and where they can be useful, can be challenging.

To contribute to the discussion each year, Big Data Quarterly presents the “Big Data 50,” a list of forward-thinking companies that are working to expand what’s possible in terms of capturing, storing, protecting, and deriving value from

data.

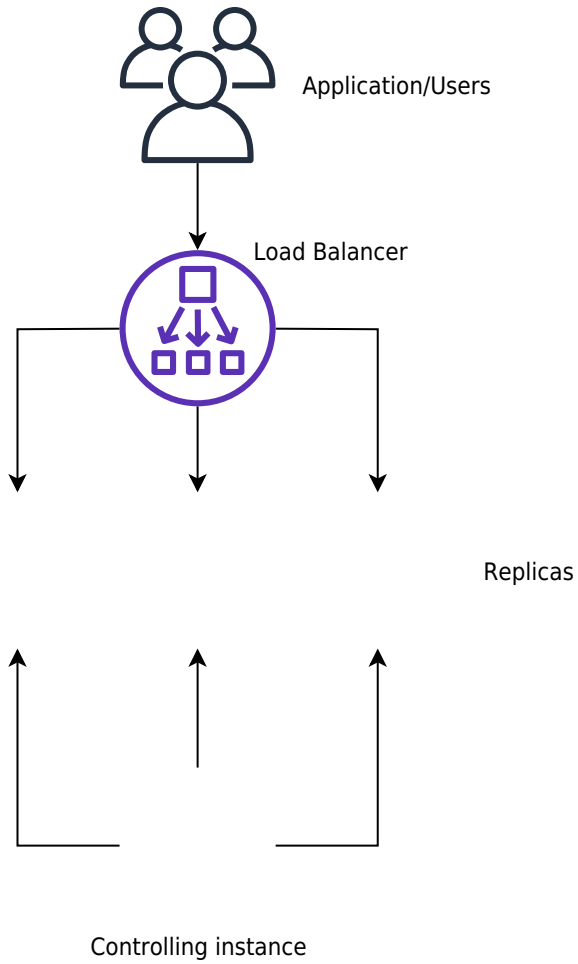
“We are honored to receive this acknowledgement for our efforts in delivering Enterprise Knowledge Graph Solutions,” said Dr. Jans Aasman, CEO, Franz Inc. “In the past year, we have seen demand for Enterprise Knowledge Graphs take off across industries along with recognition from top technology analyst firms that Knowledge Graphs provide the critical foundation for artificial intelligence applications and predictive analytics.  Our AllegroGraph Knowledge Graph Platform Solution offers a unique comprehensive approach for helping companies accelerate the creation of Enterprise Knowledge Graphs that deliver new value to their organization.”

---

# AllegroGraph Replication on Amazon’s AWS using Terraform

## Introduction

In this document we describe how to setup an AllegroGraph replication cluster on AWS using the terraform program. The cluster will have one controlling instance and a set of instances controlled by an Auto Scaling Group and reached via a Load Balancer.




Creating such a system on AWS takes a long time if done manually through their web interface. We have another document that takes you through the steps. Describing the system in terraform first takes a little time but once that's done the cluster can be started in less than five minutes.

# Steps

1. Obtain an AMI with AllegroGraph and aws-repl (our support code for aws) installed.
2. Edit the terraform file we supply to suit your needs
3. Run terraform to build the cluster

# Obtain an AMI with AllegroGraph and aws-repl

An AMI is an image of a virtual machine. You create an AMI by launching an ec2 instance using an AMI, altering the root disk

of that instance and then telling AWS to create an AMI based on your instance. You can repeat this process until you create the AMI you need.

We have a prebuild AMI with all the code installed. It uses AllegroGraph 6.5.0 and doesn’t contain a license code so it’s limited to 5 million triples. You can use this AMI to test the load balancer or you can use this image as the starting off point for building your own image.

Alternatively you start from a fresh AMI and install everything yourself as described next.

We will create an AMI to run AllegroGraph with Replication with the following features

1. When an EC2 instance running this AMI is started it starts AllegroGraph and joins the cluster of nodes serving a particular repository.
2. When the the EC2 instance is terminated the instance sends a message to the controlling instance to ensure that the terminating instance is removed from the cluster
3. If the EC2 instance is started at a particular IP address it creates the cluster and acts as the controlling instance of the cluster

This is a very simple setup but will serve many applications. For more complex needs you’ll need to write your own tools. Contact support@franz.com to discuss support options.

The choice of AMI on which to build our AMI is not important except that our scripts assume that the initial account name of the image is ec2-user. Thus we suggest that you use one of the Amazon Linux images. If you use another kind of image you’ll need to do extra work (as an example we describe below how to use a Centos AMI). Since the instance we’ll build with the AMI are used only for AllegroGraph and not for other uses there’s no point in running a different version of Linux that

you may use in your development work.

These are the steps to build an AMI:

Start an instance using an Amazon Linux AMI with EBS support.

We can’t specify the exact name of the image to start as the names change over time and depending on the region. We will usually pick one of the first images listed.

You don’t need to start a large virtual machine. A t2.micro will do.

You’ll need to specify a VPC and subnet. There should be a default VPC available. If not you’ll have to create one.

Make sure that when you specify that subnet that you want to external IP address.

Copy an agraph distribution (tar.gz format) to the ec2 instance into the home directory of ec2-user. Also copy the file aws-repl/aws-repl.tar to the home directory of ec2-user on the instance. aws-repl.tar contains scripts to support replication setup on AWS.

Extract the agraph repo in a temporary spot and run install-agraph in it, specifying the root of the agraph distribution.

I put it in /home/ec2-user/agraph

For example:

```
% mkdir tmp
% cd tmp
% tar xfz ../agraph-6.5.0-linuxamd64.64.tar.gz
% cd agraph-6.5.0
% ./install-agraph ~/agraph
```

Edit the file ~/agraph/lib/agraph.cfg and add the line

**UseMainPortForSessions yes**

This will allow sessions to be tracked through the Load Balancer.

If you have an agraph license key you should add it to the agraph.cfg file.

Unpack and install the aws-repl code:

```
% tar xf aws-repl.tar
% cd aws-repl
% sudo ./install.sh
```

You can delete aws-repl.tar but don’t delete the aws-repl directory. It will be used on startup.

Look at aws-repl/var.sh to see the parameter values. You’ll see an agraphroot parameter which should match where you installed agraph.

At this point the instance is setup.

You should go to the aws console, select this instance, and from the Action menu select “Image / Create Image”. Wait for the AMI to be built. At this time you can terminate the ec2 instance.

## Using a CentOS 7 image:

If you wish to install on top of CentOS then you’ll need additional steps. The initial user on CentOS is called ‘centos’ rather than ‘ec2-user’. In order to keep things consistent we’ll create the ec2-user account and use that for running agraph just as we do for the Amazon AMI.

ssh to the ec2 vm as centos and do the following to create the ec2-user account and to allow ssh access to it just like the centos account

```
[centos@ip-10-0-1-227 ~]$ sudo sh

sh-4.2# adduser ec2-user
```

```
sh-4.2# cp -rp .ssh ~ec2-user
sh-4.2# chown -R ec2-user ~ec2-user/.ssh
sh-4.2# exit

[centos@ip-10-0-1-227 ~]

$
```

At this point you can copy the agraph distribution to the ec2 vm. Scp to ec2-user@x.x.x.x rather than centos@x.x.x.x. Also copy the aws-repl.tar file.

The only change to the procedure is when you must run install.sh in the aws-repl directory.

The ec2-user account does not have the ability to sudo. So this command must be run

when logged in as the user centos;

```
centos@ip-10-0-1-227 ~]$ sudo sh
sh-4.2# cd ~ec2-user/aws-repl
sh-4.2# ./install.sh
+ cp joincluster /etc/rc.d/init.d
+ chkconfig --add joincluster
sh-4.2# exit

[centos@ip-10-0-1-227 ~]

$
```

# Edit the terraform file we supply to suit your needs

Edit the file agelb.tf. This file contains directives to terraform to create the cluster with load balancer. At the top are the variables you can easily change. Other values are found inside the directives and you can change those as well.

Two variables you definitely need to change are

1. “**ag-elb-ami**” – this is the name of the AMI you created in the previous step or the AMI we supply.
2. “**ssh-key**” – this is the name of the ssh key pair you want to use in the instances created.

You may wish to change the region where you want the instances built (that value is in the provider clause at the top of the file) and if you do you’ll need to change the variable “azs”.

We suggest you try building the cluster with the minimum changes to verify it works and then customize it to your liking.

# Run terraform to build the cluster

To build the cluster make sure your have an ~/.aws/config file with a default entry, such as

```
[default]
aws_access_key_id = AKIAIXXXXXXXXXXXXXX
aws_secret_access_key = o/dyrxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx
```

This is what terraform uses as credentials when it contacts AWS.

In order to use terraform the first time (or any time you change the provider clause in agelb.tf) run this command

```
% terraform init
```

Terraform will download the files appropriate for the provider you specified.

After that you can build your cluster with

```
% terraform apply
```

And watch the messages. If there are no errors terraform will wait for confirmation from you to proceed. Type yes to proceed, anything else to abort.

After terraform is finished you'll see the address of the load balancer printed.

You can make changes the agelb.tf file and again 'terraform apply ' and terraform will tell you what it needs to do to change things from how they are now to what the agelb.tf file specifies.

To delete everything terraform added type the command

% terraform destroy

And type yes when prompted.

---

# SHACL – Shapes Constraint Language in AllegroGraph

SHACL is a SHApe Constraint Language. It specifies a vocabulary (using triples) to describe the shape that data should have. The *shape* specifies things like the following simple requirements:

- How many triples with a specified subject and predicate should be in the repository (e.g. at least 1, at most 1, exactly 1).
- What the nature of the object of a triple with a specified subject and predicate should be (e.g. a string, an integer, etc.)

See the specification for more examples.

SHACL allows you to validate that your data is conforming to desired requirements.

For a given validation, the shapes are in the *Shapes*

*Graph* (where *graph* means a collection of triples) and the data to be validated is in the *Data Graph* (again, a collection of triples). The SHACL vocabularly describes how a given shape is linked to *targets* in the data and also provides a way for a Data Graph to specify the Shapes Graph that should be used for validatation. The result of a SHACL validation describes whether the Data Graph conforms to the Shapes Graph and, if it does not, describes each of the failures.

## Namespaces Used in this Document

Along with standard predefined namespaces (such as rdf: for <http://www.w3.org/1999/02/22-rdf-syntax-ns#> and rdfs: for <http://www.w3.org/2000/01/rdf-schema#>), the following are used in code and examples below:

```
prefix fr: <https://franz.com#>
prefix sh: <http://www.w3.org/ns/shacl#>
prefix franz: <https://franz.com/ns/allegrograph/6.6.0/>
```

# A Simple Example

Suppose we have a *Employee* class and for each Employee instance, there must be exactly one triple of the form

```
emp001 hasID "000-12-3456"
```

where the object is the employee's ID Number, which has the format is [3 digits]-[2 digits]-[4 digits].

This TriG file encapsulates the constraints above:

```
@prefix sh: <http://www.w3.org/ns/shacl#> .
@prefix xsd: <http://www.w3.org/2001/XMLSchema#> .

<https://franz.com#Shapes> {
 <https://franz.com#EmployeeShape>
  a sh:NodeShape ;
  sh:targetClass <https://franz.com#Employee> ;
```

```
  sh:property [
    sh:path <https://franz.com#hasID> ;
    sh:minCount 1 ;
    sh:maxCount 1 ;
    sh:datatype xsd:string ;
           sh:pattern   "^[0-9][0-9][0-9]-[0-9][0-9]-
[0-9][0-9][0-9][0-9]$" ;
  ] .
}
```

It says that for instances of fr:Employee (sh:targetClass <https://franz.com#Employee>), there must be exactly 1 triple with predicate (path) fr:hasID and the object of that triple must be a string with pattern [3 digits]-[2 digits]-[4 digits] (sh:pattern "^[0-9][0-9][0-9]-[0-9][0-9]-[0-9][0-9][0-9][0-9]$").

This TriG file defines the Employee class and some employee instances:

```
@prefix fr: <https://franz.com#> .
@prefix rdf: <http://www.w3.org/1999/02/22-rdf-syntax-ns#> .
@prefix rdfs: <http://www.w3.org/2000/01/rdf-schema#> .

{
 fr:Employee
  a rdfs:Class .
 fr:emp001
  a fr:Employee ;
  fr:hasID "000-12-3456" ;
  fr:hasID "000-77-3456" .
 fr:emp002
  a fr:Employee ;
  fr:hasID "00-56-3456" .
 fr:emp003
  a fr:Employee .
 }
```

Recalling the requirements above, we immediately see these problems with these triples:

1. *emp001* has two *hasID* triples.
2. The value of *emp002*‘s ID has the wrong format (two leading digits rather than 3).
3. *emp003* does not have a *hasID* triple.

We load the two TriG files into our repository, and end up with the following triple set. Note that all the employee triples use the default graph and the SHACL-related triples use the graph <https://franz.com#Shapes> specified in the TriG file.

| s | p | o | g |
|---|---|---|---|
| Employee | rdf:type | rdfs:Class | |
| emp001 | rdf:type | Employee | |
| emp001 | hasID | "000-12-3456" | |
| emp001 | hasID | "000-77-3456" | |
| emp002 | rdf:type | Employee | |
| emp002 | hasID | "00-56-3456" | |
| emp003 | rdf:type | Employee | |
| EmployeeShape | property | _:b7A1D241Ax1 | Shapes |
| _:b7A1D241Ax1 | pattern | "^[0-9][0-9][0-9]-[0-9][0-9]-[0-9][0-9][0-9][0-9]$" | Shapes |
| _:b7A1D241Ax1 | datatype | xs:string | Shapes |
| _:b7A1D241Ax1 | maxCount | "1" | Shapes |
| _:b7A1D241Ax1 | minCount | "1" | Shapes |
| _:b7A1D241Ax1 | path | hasID | Shapes |
| EmployeeShape | targetClass | Employee | Shapes |
| EmployeeShape | rdf:type | NodeShape | Shapes |

Now we use **agtool shacl-validate** to validate our data:

```
bin/agtool shacl-validate --data-graph default  --shapes-graph
https://franz.com#Shapes shacl-repo-1
Validation report:             Does not conform
Created:                       2019-06-27T10:24:10
Number of shapes graphs:       1
Number of data graphs:         1
Number of NodeShapes:          1
Number of focus nodes checked: 3

3 validation results:
Result:
 Focus node:           <https://franz.com#emp001>
 Path:                 <https://franz.com#hasID>
 Source Shape:         _:b7A1D241Ax1
                  Constraint                    Component:
<http://www.w3.org/ns/shacl#MaxCountConstraintComponent>
 Severity:             <http://www.w3.org/ns/shacl#Violation>
```

```
Result:
 Focus node:           <https://franz.com#emp002>
 Path:                 <https://franz.com#hasID>
 Value:                "00-56-3456"
 Source Shape:         _:b7A1D241Ax1
                  Constraint                    Component:
<http://www.w3.org/ns/shacl#PatternConstraintComponent>
 Severity:             <http://www.w3.org/ns/shacl#Violation>

Result:
 Focus node:           <https://franz.com#emp003>
 Path:                 <https://franz.com#hasID>
 Source Shape:         _:b7A1D241Ax1
                  Constraint                    Component:
<http://www.w3.org/ns/shacl#MinCountConstraintComponent>
 Severity:             <http://www.w3.org/ns/shacl#Violation>
```

The validation fails with the problems listed above. The **Focus node** is the subject of a triple that did not conform. **Path** is the predicate or a property path (predicates in this example). **Value** is the offending value. **Source Shape** is the shape that established the constraint (you must look at the shape triples to see exactly what **Source Shape** is requiring).

We revise our employee data with the following SPARQL expresssion, deleting one of the emp001 triples, deleting the emp002 triple and adding a new one with the correct format, and adding an emp003 triple.

```
prefix fr: <https://franz.com#>

DELETE DATA {fr:emp002 fr:hasID "00-56-3456" } ;

INSERT DATA {fr:emp002 fr:hasID "000-14-1772" } ;

DELETE DATA {fr:emp001 fr:hasID "000-77-3456" } ;

INSERT DATA {fr:emp003 fr:hasID "000-54-9662" } ;
```

Now our employee triples are

| s | p | o | g |
|---|---|---|---|
| emp002 | hasID | "000-14-1772" | |
| emp003 | hasID | "000-54-9662" | |
| Employee | rdf:type | rdfs:Class | |
| emp001 | rdf:type | Employee | |
| emp001 | hasID | "000-12-3456" | |
| emp002 | rdf:type | Employee | |
| emp003 | rdf:type | Employee | |

We run the validation again and are told our data conforms:

```
% bin/agtool shacl-validate --data-graph default  --shapes-graph https://franz.com#Shapes shacl-repo-1
Validation report:             Conforms
Created:                       2019-06-27T10:32:19
Number of shapes graphs:       1
Number of data graphs:         1
Number of NodeShapes:          1
Number of focus nodes checked: 3
```

When we refer to this example in the remainder of this document, it is to the un-updated (incorrect) triples.

# SHACL API

The example above illustrates the SHACL steps:

1. Have a data set with triples that should conform to a shape
2. Have SHACL triples that express the desired shape
3. Run SHACL validation to determine if the data conforms

Note that SHACL validation does not modify the data being validated. Once you have the conformance report, you must modify the data to fix the conformance problems and then rerun the validation test.

The main entry point to the API is **agtool shacl-validate**. It takes various options and has several output choices. Online help for **agtool shacl-validate** is displayed by running agtool shacl-validate --help.

In order to validate triples, the system must know:

1. What tripes to examine
2. What rules (SHACL triples) to use
3. What to do with the results

## Specifying what triples to examine

Two arguments to **agtool shacl-validate** specify the triples to evaluate: --data-graph and --focus-node. Each can be specified multiple times.

- The --data-graph argument specifies the graph value for triples to be examined. Its value must be an IRI or default. Only triples in the specified graphs will be examined. default specifies the default graph. It is also the default value of the --data-graph argument. If no value is specified for --data-graph, only triples in the default graph will be examined. If a value for --data-graph is specified, triples in the default graph will only be examined if --data-graph default is also specified.
- The --focus-node argument specifies IRIs which are subjects of triples. If this argument is specified, only triples with these subjects will be examined. To be examined, triples must also have graph values specified by --data-graph arguments. --focus-node does not have a default value. If unspecified, all triples in the specified data graphs will be examined. This argument can be specified multiple times.

The --data-graph argument was used in the simple example above. Here is how the --focus-node argument can be used to restrict validation to triples with subjects <https://franz.com#emp002>and <https://franz.com#emp003> and to ignore triples with subject <https://franz.com#emp001> (applying **agtool shacl-validate** to the orignal non-conformant data):

```
% bin/agtool shacl-validate --data-graph default  \
```

```
  --shapes-graph https://franz.com#Shapes \
  --focus-node https://franz.com#emp003 \
  --focus-node https://franz.com#emp002 shacl-repo-1
Validation report:             Does not conform
Created:                       2019-06-27T11:37:49
Number of shapes graphs:       1
Number of data graphs:         1
Number of NodeShapes:          1
Number of focus nodes checked: 2

2 validation results:
Result:
 Focus node:           <https://franz.com#emp003>
 Path:                 <https://franz.com#hasID>
 Source Shape:         _:b7A1D241Ax2
                  Constraint                  Component:
<http://www.w3.org/ns/shacl#MinCountConstraintComponent>
 Severity:             <http://www.w3.org/ns/shacl#Violation>

Result:
 Focus node:           <https://franz.com#emp002>
 Path:                 <https://franz.com#hasID>
 Value:                "00-56-3456"
 Source Shape:         _:b7A1D241Ax2
                  Constraint                  Component:
<http://www.w3.org/ns/shacl#PatternConstraintComponent>
 Severity:             <http://www.w3.org/ns/shacl#Violation>
```

## Specifying What Shape Triples to Use

Two arguments to **agtool shacl-validate**, analogous to the two arguments for data described above, specify Shape triples to use. Further, following the SHACL spec, data triples with predicate <http://www.w3.org/ns/shacl#shapeGraph> also specify graphs containing Shape triples to be used.

The arguments to **agtool shacl-validate** are the following. Each may be specified multiple times.

- The --shapes-graph argument specifies the graph value

for shape triples to be used for SHACL validation. Its value must be an IRI or default. default specifies the default graph. The --shapes-graph argument has no default value. If unspecified, graphs specified by data triples with the <http://www.w3.org/ns/shacl#shapeGraph> predicate will be used (they are used whether or not --shapes-graph has a value). If --shapes-graph has no value and there are no data triples with the <http://www.w3.org/ns/shacl#shapeGraph> predicate, the data graphs are used for shape graphs. (Shape triples have a known format and so can be identified among the data triples.)

- The --shape argument specifies IRIs which are subjects of shape nodes. If this argument is specified, only shape triples with these subjects and subsiduary triples to these will be used for validation. To be included, the triples must also have graph values specified by the --shapes-graph arguments or specified by a data triple with the <http://www.w3.org/ns/shacl#shapeGraph> predicate. --shape does not have a default value. If unspecified, all shapes in the shapes graphs will be used.

# Other APIs

There is a lisp API using the function validate-data-graph, defined next:

validate-data-graphdb &key data-graph-iri/s shapes-graph-iri/s shape/s focus-node/s verbose conformance-only? function

Perform SHACL validation and return a validation-report structure.

The validation uses data-graph-iri/s to construct the

dataGraph. This can be a single IRI, a list of IRIs or NIL, in which case the default graph will be used. The shapesGraph can be specified using the shapes-graph-iri/s parameter which can also be a single IRI or a list of IRIs. If shape-graph-iri/s is not specified, the SHACL processor will first look to create the shapesGraph by finding triples with the predicate sh:shapeGraph in the dataGraph. If there are no such triples, then the shapesGraph will be assumed to be the same as the dataGraph.

Validation can be restricted to particular shapes and focus nodes using the shape/s and focus-node/s parameters. Each of these can be an IRI or list of IRIs.

If conformance-only? is true, then validation will stop as soon as any validation failures are detected.

You can use validation-report-conforms-p to see whether or not the dataGraph conforms to the shapesGraph (possibly restricted to just particular shape/s and focus-node/s).

The function validation-report-conforms-p returns t or nil as the validation struct returned by validate-data-graph does or does not conform.

validation-report-conforms-preport
function

Returns t or nil to indicate whether or not REPORT (a validation-report struct) indicates that validation conformed. There is also a REST API. See HTTP reference.

# Validation Output

The simple example above and the SHACL examples below show output from **agtool validate-shacl**. There are various output formats, specified by the --output option. Those examples use the plain format, which means printing results descriptively.

Other choices include json, trig, trix, turtle, nquads, rdf-n3, rdf/xml, and ntriples. Here are the simple example (uncorrected) results using ntriples output:

```
% bin/agtool shacl-validate --output ntriples --data-graph default --shapes-graph https://franz.com#Shapes shacl-repo-1

_:b271983AAx1 <http://www.w3.org/1999/02/22-rdf-syntax-ns#type> <http://www.w3.org/ns/shacl#ValidationReport> .
_:b271983AAx1 <http://www.w3.org/ns/shacl#conforms> "false"^^<http://www.w3.org/2001/XMLSchema#boolean> .
_:b271983AAx1 <http://purl.org/dc/terms/created> "2019-07-01T18:26:03"^^<http://www.w3.org/2001/XMLSchema#dateTime> .
_:b271983AAx1 <http://www.w3.org/ns/shacl#result> _:b271983AAx2 .
_:b271983AAx2 <http://www.w3.org/1999/02/22-rdf-syntax-ns#type> <http://www.w3.org/ns/shacl#ValidationResult> .
_:b271983AAx2 <http://www.w3.org/ns/shacl#focusNode> <https://franz.com#emp001> .
_:b271983AAx2 <http://www.w3.org/ns/shacl#resultPath> <https://franz.com#hasID> .
_:b271983AAx2 <http://www.w3.org/ns/shacl#resultSeverity> <http://www.w3.org/ns/shacl#Violation> .
_:b271983AAx2 <http://www.w3.org/ns/shacl#sourceConstraintComponent> <http://www.w3.org/ns/shacl#MaxCountConstraintComponent> .
_:b271983AAx2 <http://www.w3.org/ns/shacl#sourceShape> _:b271983AAx3 .
_:b271983AAx1 <http://www.w3.org/ns/shacl#result> _:b271983AAx4 .
_:b271983AAx4 <http://www.w3.org/1999/02/22-rdf-syntax-ns#type> <http://www.w3.org/ns/shacl#ValidationResult> .
_:b271983AAx4 <http://www.w3.org/ns/shacl#focusNode> <https://franz.com#emp002> .
_:b271983AAx4 <http://www.w3.org/ns/shacl#resultPath> <https://franz.com#hasID> .
_:b271983AAx4 <http://www.w3.org/ns/shacl#resultSeverity>
```

```
<http://www.w3.org/ns/shacl#Violation> .
_:b271983AAx4
<http://www.w3.org/ns/shacl#sourceConstraintComponent>
<http://www.w3.org/ns/shacl#PatternConstraintComponent> .
_:b271983AAx4    <http://www.w3.org/ns/shacl#sourceShape>
_:b271983AAx3 .
_:b271983AAx4 <http://www.w3.org/ns/shacl#value> "00-56-3456"
.
_:b271983AAx1      <http://www.w3.org/ns/shacl#result>
_:b271983AAx5 .
_:b271983AAx5
<http://www.w3.org/1999/02/22-rdf-syntax-ns#type>
<http://www.w3.org/ns/shacl#ValidationResult> .
_:b271983AAx5     <http://www.w3.org/ns/shacl#focusNode>
<https://franz.com#emp003> .
_:b271983AAx5     <http://www.w3.org/ns/shacl#resultPath>
<https://franz.com#hasID> .
_:b271983AAx5   <http://www.w3.org/ns/shacl#resultSeverity>
<http://www.w3.org/ns/shacl#Violation> .
_:b271983AAx5
<http://www.w3.org/ns/shacl#sourceConstraintComponent>
<http://www.w3.org/ns/shacl#MinCountConstraintComponent> .
_:b271983AAx5    <http://www.w3.org/ns/shacl#sourceShape>
_:b271983AAx3 .
```

You can have the triples added to the repository by specifying the --add-to-repo option true.

In the plain output information is provided about how many data graphs are examined, how many shape graphs were specified and node shapes are found, and how many focus nodes are checked. If zero focus nodes are checked, that is likely not what you want and something has gone wrong. Here we mis-spell the name of the shape graph (https://franz.com#shapes instead of https://franz.com#Shapes) and get 0 focus nodes checked:

```
% bin/agtool shacl-validate --data-graph default --shapes-
graph https://franz.com#shapes shacl-repo-1
Validation report:            Conforms
Created:                      2019-06-28T10:34:22
Number of shapes graphs:      1
```

```
Number of data graphs:         1
Number of NodeShapes:          0
Number of focus nodes checked: 0
```

# SPARQL integration

There are two sets of magic properties defined: one checks for basic conformance and the other produces validation reports as triples:

- ?valid franz:shaclConforms ( ?dataGraph [ ?shapesGraph ] )
- ?valid franz:shaclFocusNodeConforms1 ( ?dataGraph ?nodeOrNodeCollection )
- ?valid franz:shaclFocusNodeConforms2 ( ?dataGraph ?shapesGraph ?nodeOrNodeCollection )
- ?valid franz:shaclShapeConforms1 ( ?dataGraph ?shapeOrShapeCollection [ ?nodeOrNodeCollection ] )
- ?valid franz:shaclShapeConforms2 ( ?dataGraph ?shapesGraph ?shapeOrShapeCollection [ ?nodeOrNodeCollection ] )
- (?s ?p ?o) franz:shaclValidationReport ( ?dataGraph [ ?shapesGraph ] )
- (?s ?p ?o) franz:shaclFocusNodeValidationReport1 ( ?dataGraph ?nodeOrNodeCollection ) .
- (?s ?p ?o) franz:shaclFocusNodeValidationReport2 ( ?dataGraph ?shapesGraph ?nodeOrNodeCollection ) .
- (?s ?p ?o) franz:shaclShapeValidationReport1 ( ?dataGraph ?shapeOrShapeCollection [ ?nodeOrNodeCollection ] ) .
- (?s ?p ?o) franz:shaclShapeValidationReport2 ( ?dataGraph ?shapesGraph ?shapeOrShapeCollection [ ?nodeOrNodeCollection ] ) .

In all of the above ?dataGraph and ?shapesGraph can be IRIs, the literal ‘default’, or a variable that is bound to a SPARQL collection (list or set) that was previously created with a

function
like https://franz.com/ns/allegrograph/6.5.0/fn#makeSPARQLList or https://franz.com/ns/allegrograph/6.5.0/fn#lookupRdfList. If a collection is used, then the SHACL processor will create a temporary RDF merge of all of the graphs in it to produce the data graph or the shapes graph.

Similarly, ?shapeOrShapeCollection and ?nodeOrNodeCollection can be bound to an IRI or a SPARQL collection. If a collection is used, then it must be bound to a list of IRIs. The SHACL processor will restrict validation to the shape(s) and focus node(s) (i.e. nodes that should be validated) specified.

The shapesGraph argument is optional in both of the shaclConforms and shaclValidationReport magic properties. If the shapesGraph is not specified, then the shapesGraph will be created by following triples in the dataGraph that use the sh:shapesGraph predicate. If there are no such triples, then the shapesGraph will be the same as the dataGraph.

For example, the following SPARQL expression

```
construct { ?s ?p ?o } where {
  # form a collection of focusNodes
bind(<https://franz.com/ns/allegrograph/6.6.0/fn#makeSPARQLLis
t>(
    <http://Journal1/1942/Article25>,
    <http://Journal1/1943>) as ?nodes)
                          (?s             ?p             ?o)
<https://franz.com/ns/allegrograph/6.6.0/shaclShapeValidationR
eport1>
    ('default' <ex://franz.com/documentShape1> ?nodes) .
}
```

would use the default graph as the Data Graph and the Shapes Graph and then validate two focus nodes against the shape <ex://franz.com/documentShape1>.

# SHACL Example

We build on our simple example above. Start with a fresh repository so triples from the simple example do not interfere with this example.

We start with a TriG file with various shapes defined on some classes.

```
@prefix sh: <http://www.w3.org/ns/shacl#> .
@prefix xsd: <http://www.w3.org/2001/XMLSchema#> .
@prefix fr: <https://franz.com#> .
@prefix rdf: <http://www.w3.org/1999/02/22-rdf-syntax-ns#> .
@prefix rdfs: <http://www.w3.org/2000/01/rdf-schema#> .

<https://franz.com#ShapesGraph> {
fr:EmployeeShape
   a sh:NodeShape ;
   sh:targetClass fr:Employee ;
   sh:property [
     ## Every employee must have exactly one ID
     sh:path fr:hasID ;
     sh:minCount 1 ;
     sh:maxCount 1 ;
     sh:datatype xsd:string ;
            sh:pattern  "^[0-9][0-9][0-9]-[0-9][0-9]-
[0-9][0-9][0-9][0-9]$" ;
    ] ;
   sh:property [
     ## Every employee is a manager or a worker
     sh:path fr:employeeType ;
     sh:minCount 1 ;
     sh:maxCount 1 ;
     sh:datatype xsd:string ;
     sh:in ("Manager" "Worker") ;
    ] ;
    sh:property [
      ## If birthyear supplied, must be 2001 or before
      sh:path fr:birthYear ;
      sh:maxInclusive 2001 ;
      sh:datatype xsd:integer ;
```

```
] ;
sh:property [
  ## Must have a title, may have more than one
  sh:path fr:hasTitle ;
  sh:datatype xsd:string ;
  sh:minCount 1 ;
] ;

sh:or (
  ## The President does not have a supervisor
  [
    sh:path fr:hasTitle ;
    sh:hasValue "President" ;
  ]
  [
   ## Must have a supervisor
     sh:path fr:hasSupervisor ;
     sh:minCount 1 ;
     sh:maxCount 1 ;
     sh:class fr:Employee ;
  ]
  ) ;

sh:or (
  # Every employee must either have a wage or a salary
  [
   sh:path fr:hasSalary ;
   sh:datatype xsd:integer ;
   sh:minInclusive 3000 ;
   sh:minCount 1 ;
   sh:maxCount 1 ;
  ]
  [
   sh:path fr:hasWage ;
   sh:datatype xsd:decimal ;
   sh:minExclusive 15.00 ;
   sh:minCount 1 ;
   sh:maxCount 1 ;
  ]
)
.
```

```
   }
```

This file says the following about instances of the class fr:Employee:

1. Every employee must have exactly one ID (object of fr:hasID), a string of the form NNN-NN-NNNN where the Ns are digits (this is the simple example requirement).
2. Every employee must have exactly one fr:employeeType triple with value either “Manager” or “Worker”.
3. Employees may have a fr:birthYear triple, and if so, the value must be 2001 or earlier.
4. Employees must have a fr:hasTitle and may have more than one.
5. All employees except the one with title “President” must have a supervisor (specified with fr:hasSupervisor).
6. Every employee must either have a wage (a decimal specifying hourly pay, greater than 15.00) or a salary (an integer specifying monthly pay, greater than or equal to 3000).

Here is some employee data:

```
@prefix fr: <https://franz.com#> .
@prefix rdf: <http://www.w3.org/1999/02/22-rdf-syntax-ns#> .
@prefix rdfs: <http://www.w3.org/2000/01/rdf-schema#> .
@prefix xsd: <http://www.w3.org/2001/XMLSchema#> .

{
 fr:Employee
  a rdfs:Class .

 fr:emp001
  a fr:Employee ;
  fr:hasID "000-12-3456" ;
  fr:hasTitle "President" ;
  fr:employeeType "Manager" ;
  fr:birthYear "1953"^^xsd:integer ;
```

```
 fr:hasSalary "10000"^^xsd:integer .

fr:emp002
 a fr:Employee ;
 fr:hasID "000-56-3456" ;
 fr:hasTitle "Foreman" ;
 fr:employeeType "Worker" ;
 fr:birthYear "1966"^^xsd:integer ;
 fr:hasSupervisor fr:emp003 ;
 fr:hasWage "20.20"^^xsd:decimal .

fr:emp003
 a fr:Employee ;
 fr:hasID "000-77-3232" ;
 fr:hasTitle "Production Manager" ;
 fr:employeeType "Manager" ;
 fr:birthYear "1968"^^xsd:integer ;
 fr:hasSupervisor fr:emp001 ;
 fr:hasSalary "4000"^^xsd:integer .

fr:emp004
 a fr:Employee ;
 fr:hasID "000-88-3456" ;
 fr:hasTitle "Fitter" ;
 fr:employeeType "Worker" ;
 fr:birthYear "1979"^^xsd:integer ;
 fr:hasSupervisor fr:emp002 ;
 fr:hasWage "17.20"^^xsd:decimal .

fr:emp005
 a fr:Employee ;
 fr:hasID "000-99-3492" ;
 fr:hasTitle "Fitter" ;
 fr:employeeType "Worker" ;
 fr:birthYear "2000"^^xsd:integer ;
 fr:hasWage "17.20"^^xsd:decimal .

fr:emp006
 a fr:Employee ;
 fr:hasID "000-78-5592" ;
 fr:hasTitle "Filer" ;
```

```
  fr:employeeType "Intern" ;
  fr:birthYear "2003"^^xsd:integer ;
  fr:hasSupervisor fr:emp002 ;
  fr:hasWage "14.20"^^xsd:decimal .

 fr:emp007
  a fr:Employee ;
  fr:hasID "000-77-3232" ;
  fr:hasTitle "Sales Manager" ;
  fr:hasTitle "Vice President" ;
  fr:employeeType "Manager" ;
  fr:birthYear "1962"^^xsd:integer ;
  fr:hasSupervisor fr:emp001 ;
  fr:hasSalary "7000"^^xsd:integer .
 }
```

Comparing these data with the requirements, we see these problems:

1. emp005 does not have a supervisor.
2. emp006 is pretty messed up, with (1) employeeType “Intern”, not an allowed value, (2) a birthYear (2003) later than the required maximum of 2001, and (3) a wage (14.40) less than the minimum (15.00).

Otherwise the data seems OK.

We load these two TriG files into an emply repository (which we have named **shacl-repo-2**). We specify the default graph for the data and the https://franz.com#ShapesGraph for the shapes. (Though not required, it is a good idea to specify a graph for shape data as it makes it easy to delete and reload shapes while developing.) We have 101 triples, 49 data and 52 shape. Then we run **agtool shacl-validate**:

```
% bin/agtool shacl-validate --shapes-graph https://franz.com#ShapesGraph --data-graph default shacl-repo-2
```

There are four violations, as expected, one for emp005 and three for emp006.

```
Validation report:             Does not conform
Created:                       2019-07-03T11:35:27
Number of shapes graphs:       1
Number of data graphs:         1
Number of NodeShapes:          1
Number of focus nodes checked: 7

4 validation results:
Result:
 Focus node:           <https://franz.com#emp005>
 Value:                <https://franz.com#emp005>
 Source Shape:         <https://franz.com#EmployeeShape>
                   Constraint                  Component:
<https://www.w3.org/ns/shacl#OrConstraintComponent>
 Severity:             <https://www.w3.org/ns/shacl#Violation>

Result:
 Focus node:           <https://franz.com#emp006>
 Path:                 <https://franz.com#employeeType>
 Value:                "Intern"
 Source Shape:         _:b19D062B9x221
                   Constraint                  Component:
<http://www.w3.org/ns/shacl#InConstraintComponent>
 Severity:             <http://www.w3.org/ns/shacl#Violation>

Result:
 Focus node:           <https://franz.com#emp006>
 Path:                 <https://franz.com#birthYear>
                                                 Value:
"2003"^^<http://www.w3.org/2001/XMLSchema#integer>
 Source Shape:         _:b19D062B9x225
                   Constraint                  Component:
<http://www.w3.org/ns/shacl#MaxInclusiveConstraintComponent>
 Severity:             <http://www.w3.org/ns/shacl#Violation>

Result:
 Focus node:           <https://franz.com#emp006>
 Value:                <https://franz.com#emp006>
 Source Shape:         <https://franz.com#EmployeeShape>
                   Constraint                  Component:
<http://www.w3.org/ns/shacl#OrConstraintComponent>
```

Severity: <http://www.w3.org/ns/shacl#Violation>

Fixing the data is left as an exercise for the reader.

---

# What is the Answer to AI Model Risk Management?

**Algorithm-XLab – March 2019**

**Franz CEO Dr. Jans Aasman Explains how to manage AI Modelling Risks.**

AI model risk management has moved to the forefront of contemporary concerns for statistical Artificial Intelligence, perhaps even displacing the notion of ethics in this regard because of the immediate, undesirable repercussions of tenuous machine learning and deep learning models.

AI model risk management requires taking steps to ensure that the models used in artificial applications produce results that are unbiased, equitable, and repeatable.


The objective is to ensure that given the same inputs, they produce the same outputs.

If organizations cannot prove how they got the results of AI risk models, or have results that are discriminatory, they are subject to regulatory scrutiny and penalties.

Strict regulations throughout the financial services industry in the United Statesand Europe require governing, validating, re-validating, and demonstrating the transparency of models for financial products.

There’s a growing cry for these standards in other heavily regulated industries such as healthcare, while the burgeoning Fair, Accountable, Transparent movementtypifies the horizontal demand to account for machine learning models’ results.

AI model risk management is particularly critical in finance.

Financial organizations must be able to demonstrate how they derived the offering of any financial product or service for specific customers.

When deploying AI risk models for these purposes, they must ensure they can explain (to customers and regulators) the results that determined those offers.

**Read the full article at Algorithm-XLab.**

---

# Why Is JSON-LD Important To Businesses?

**Forbes – February 2019**

Although you may not have heard of JavaScript Object Notation Linked Data (JSON-LD), it is already affecting your business. Search engine giant Google has mentioned JSON-LD as a preferred means of adding structured data to webpages to make them considerably easier to parse for more accurate search engine results. The Google use case is indicative of the

larger capacity for JSON-LD to increase web traffic for sites and better guide users to the results they want.


Expectations are high for JSON-LD, and with good reason. It effectively delivers the many benefits of JSON, a lightweight data interchange format, into the linked data world. Linked data is the technological approach supporting the World Wide Web and one of the most effective means of sharing data ever devised.

In addition, the growing number of enterprise knowledge graphs fully exploit the potential of JSON-LD as it enables organizations to readily access data stored in document formats and a variety of semi-structured and unstructured data as well. By using this technology to link internal and external data, knowledge graphs exemplify the linked data approach underpinning the growing adoption of JSON-LD – and the demonstrable, recurring business value that linked data consistently provides.

Read the full article at Forbes.

---

# JSON-LD: A Method of Encoding Linked Data That Adds Meaning

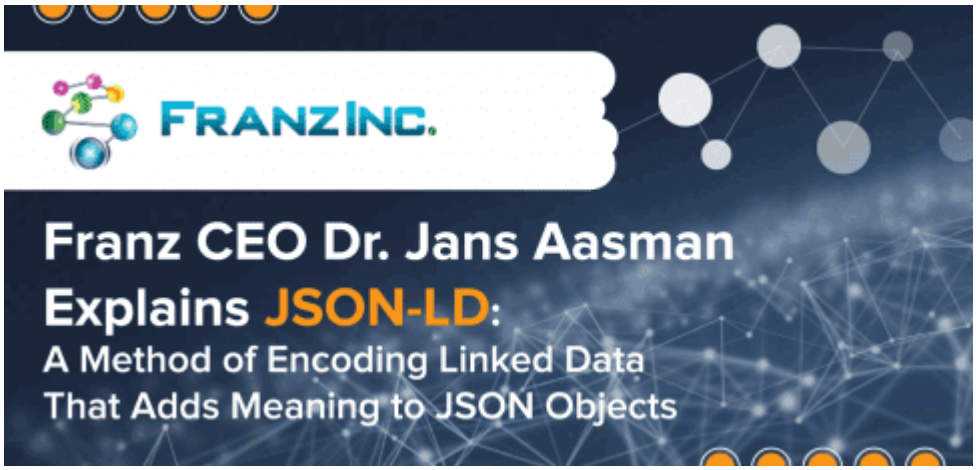
# to JSON Objects

**Hosting Advice – February 2019**

Franz CEO Dr. Jans Aasman Explains JSON-LD: A Method of Encoding Linked Data That Adds Meaning to JSON Objects.

JSON-LD, a method of presenting structured Schema.org data to search engines and other parties, helps organize and connect information online. As Dr. Jans Aasman, CEO of Franz Inc. told us, the data-interchange format has far-reaching implications, from standardizing the ecommerce and healthcare industries to building knowledge graphs. With technologies like AllegroGraph helping to convert complex data into insights, JSON-LD is being put to use in a number of ways.

Read the full article at Hosting Advice.



# Unraveling the Quandary of Access Layer versus Storage Layer Security

**InfoSecurity – February 2019**

Dr. Jans Aasman was quoted in this article about how AllegroGraph's Triple Attributes provide Storage Layer Security.

With horizontal standards such as the General Data Protection Regulation (GDPR) and vertical mandates like the Fair Credit Reporting Act increasing in scope and number, information security is impacted by regulatory compliance more than ever.

Organizations frequently decide between concentrating protection at the access layer via role-based security filtering, or at the storage layer with methods like encryption, masking, and tokenization.

The argument is that the former underpins data governance policy and regulatory compliance by restricting data access according to department or organizational role. However, the latter's perceived as providing more granular security implemented at the data layer.

> ***A hybrid of access based security and security at the data layer—implemented by triple attributes—can counteract the weakness of each approach with the other's strength, resulting in information security that Franz CEO Jans Aasman characterized as "fine-grained and flexible enough" for any***

*regulatory requirements or security model.*

*The security provided by this semantic technology is considerably enhanced by the addition of* ***key-value pairs*** *as JSON objects, which can be arbitrarily assigned to triples within databases. These key-value pairs provide a second security mechanism "embedded in the storage, so you cannot cheat," Aasman remarked.*

*When implementing HIPPA standards with triple attributes, "even if you're a doctor, you can only see a patient record if all your other attributes are okay," Aasman mentioned.*

*"We're talking about a very flexible mechanism where we can add any combination of key-value pairs to any triples, and have a very flexible language to specify how to use that to create flexible security models," Aasman said.*

Read the full article at InfoSecurity.

---

# Semantic Web and Semantic Technology Trends in 2019

**Dataversity – January 2019**

**What to expect of Semantic Web and other Semantic Technologies**

**in 2019? Quite a bit. DATAVERSITY engaged with leaders in the space to get their thoughts on how Semantic Technologies will have an impact on multiple areas.**

**Dr. Jans Aasman, CEO of Franz Inc. was quoted several times in the article:**

Among the semantic-driven AI ventures next year will be those that relate to the healthcare space, says Dr. Jans Aasman, CEO of Semantic Web technology company Franz, Inc:

"In the last two years some of the technologies were starting to get used in production," he says. "In 2019 we will see a ramp-up of the number of AI applications that will help save lives by providing early warning signs for impending diseases. Some diseases will be predicted years in advance by using genetic patient data to understand future biological issues, like the likelihood of cancerous mutations – and start preventive therapies before the disease takes hold."

If that's not enough, how about digital immortality via AI Knowledge Graphs, where an interactive voice system will bring public figures in contact with anyone in the real world? "We'll see the first examples of Digital Immortality in 2019 in the form of AI Digital Personas for public figures," says Aasman, whose company is a partner in the Noam Chomsky Knowledge Graph:

"The combination of Artificial Intelligence and Semantic Knowledge Graphs will be used to transform the works of scientists, technologists, politicians, and scholars like Noam Chomsky into an interactive response system that uses the person's actual voice to answer questions," he comments.

"AI Digital Personas will dynamically link information from various sources – such as books, research papers, notes and media interviews – and turn the disparate information into a

knowledge system that people can interact with digitally." These AI Digital Personas could also be used while the person is still alive to broaden the accessibility of their expertise.

On the point of the future of graph visualization apps, Aasman notes that:

"Most graph visualization applications show network diagrams in only two dimensions, but it is unnatural to manipulate graphs on a flat computer screen in 2D. Modern R virtual reality will add at least two dimensions to graph visualization, which will create a more natural way to manipulate complex graphs by incorporating more depth and temporal unfolding to understand information within a time perspective."

**Read the full article at Dataversity.**

---

# Solving Knowledge Graph Data Prep with Standards

**Dataversity – December 2018**

There's a general consensus throughout the data ecosystem that Data Preparation is the most substantial barrier to capitalizing on data-driven processes. Whether organizations are embarking on Data Science initiatives or simply feeding any assortment of enterprise applications, the cleansing, classifying, mapping, modeling, transforming, and integrating

of data is the most time honored (and time consuming) aspect of this process.

Approximately 80 percent of the work of data scientists is mired in Data Preparation, leaving roughly 20 percent of their jobs to actually exploiting data. Moreover, the contemporary focus on external sources, Big Data, social and mobile technologies has exploded the presence of semi-structured and unstructured data, which accounts for nearly 80 percent of today’s data and further slows the preparation processes.

Read the full article at Dataversity.