Pivotal

Federated Queries with Greenplum and PXF

Alexander Denissov

Software Architect

April 2018

Pivotal Greenplum®

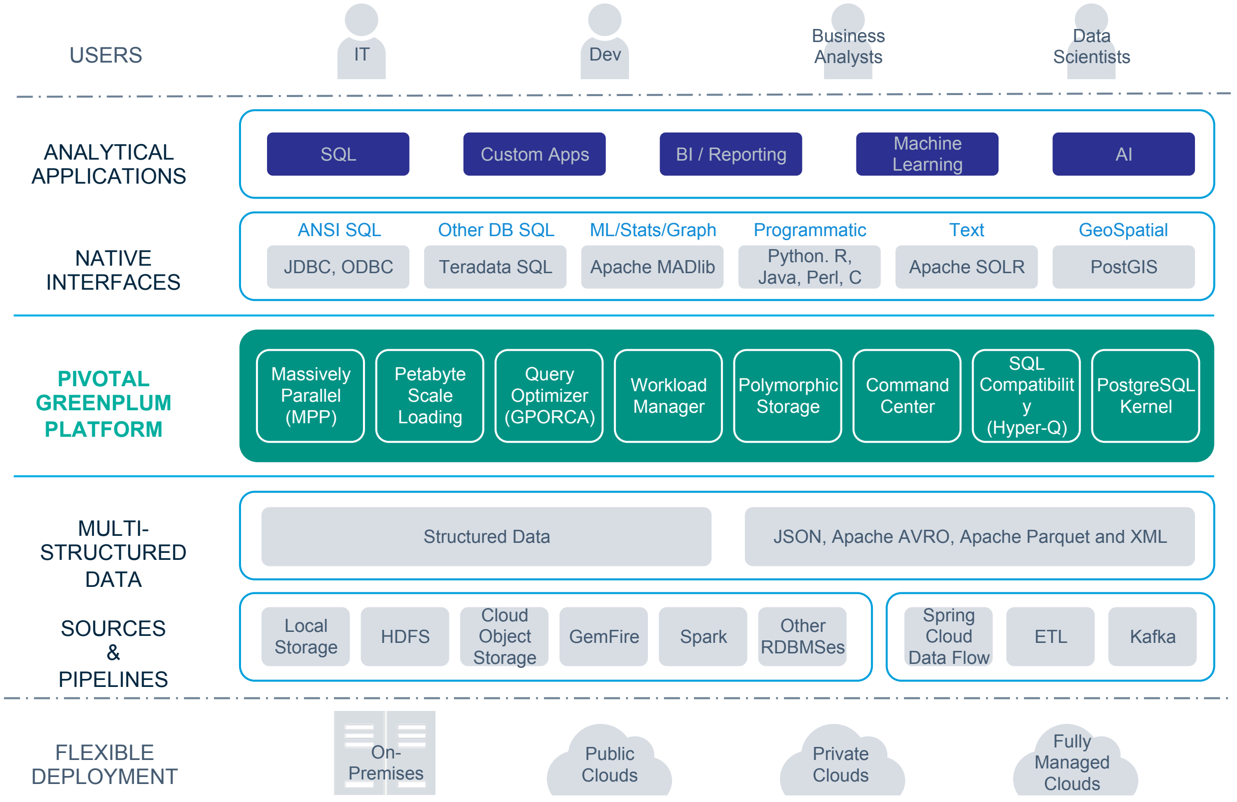
© Copyright 2017 Pivotal Software, Inc. All rights Reserved. Version 1.0

Agenda

- Introduction to Federated Queries
- Federation Use Cases
- Greenplum External Tables
- PXF Architecture
- PXF Connectors and Profiles
- Advanced Topics
- Q+A

Data Platform for Analytics

Pivotal Greenplum®



USERS: IT, Dev, Business Analysts, Data Scientists

ANALYTICAL APPLICATIONS: SQL, Custom Apps, BI / Reporting, Machine Learning, AI

NATIVE INTERFACES:

ANSI SQL	Other DB SQL	ML/Stats/Graph	Programmatic	Text	GeoSpatial
JDBC, ODBC	Teradata SQL	Apache MADlib	Python. R, Java, Perl, C	Apache SOLR	PostGIS

PIVOTAL GREENPLUM PLATFORM: Massively Parallel (MPP), Petabyte Scale Loading, Query Optimizer (GPORCA), Workload Manager, Polymorphic Storage, Command Center, SQL Compatibilit y (Hyper-Q), PostgreSQL Kernel

MULTI-STRUCTURED DATA: Structured Data; JSON, Apache AVRO, Apache Parquet and XML

SOURCES & PIPELINES: Local Storage, HDFS, Cloud Object Storage, GemFire, Spark, Other RDBMSes; Spring Cloud Data Flow, ETL, Kafka

FLEXIBLE DEPLOYMENT: On-Premises, Public Clouds, Private Clouds, Fully Managed Clouds

The world's first open-source massively parallel processing (MPP) data platform for advanced analytics

Based on PostgreSQL

Developed since early 2000s

Open sourced in 2015

SQL 2003 compliant

Advanced cost-based optimizer

ACID transactions guarantees

Pivotal

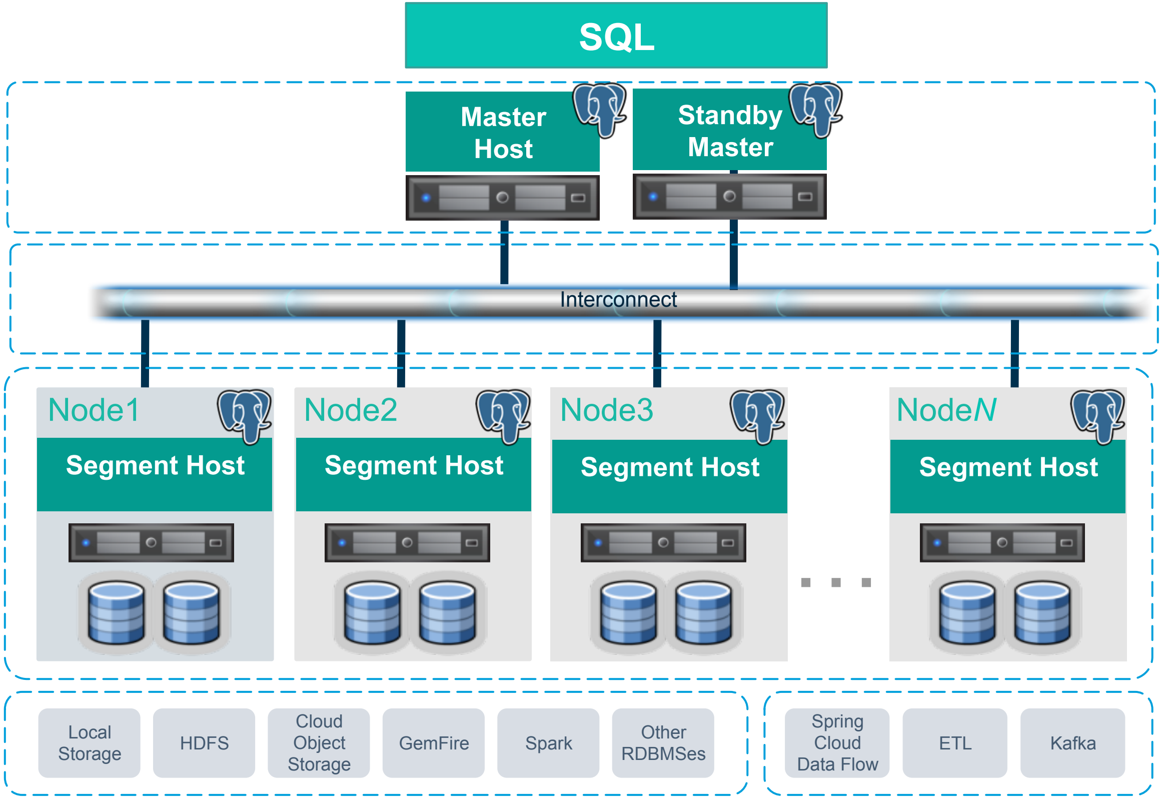
Greenplum = Massively Parallel Postgres for Analytics

Master Servers
Query planning and dispatch

Interconnect

Segment Servers
Query processing and data storage

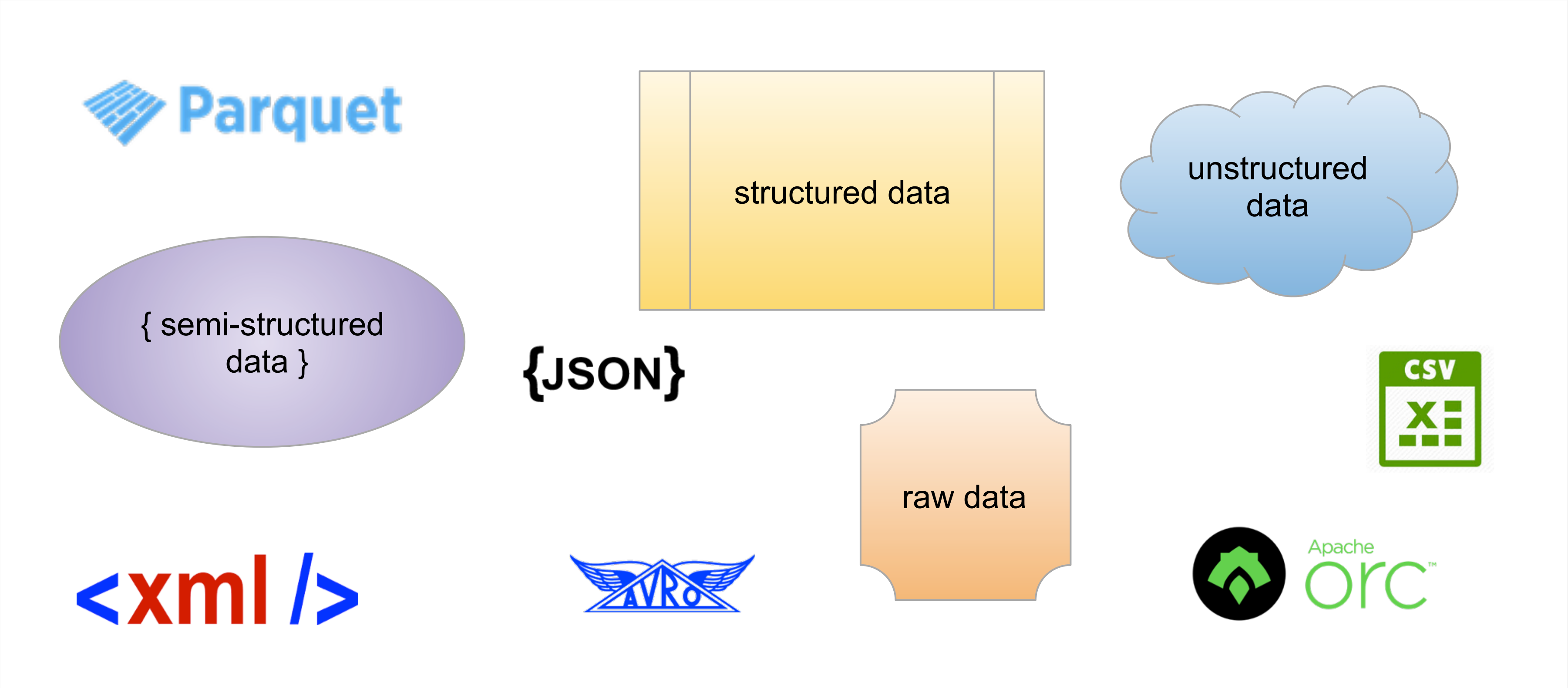
External Sources & Pipelines
Parallel loading and streaming



Pivotal

Pivotal Greenplum

Modern Enterprise : heterogeneous data formats



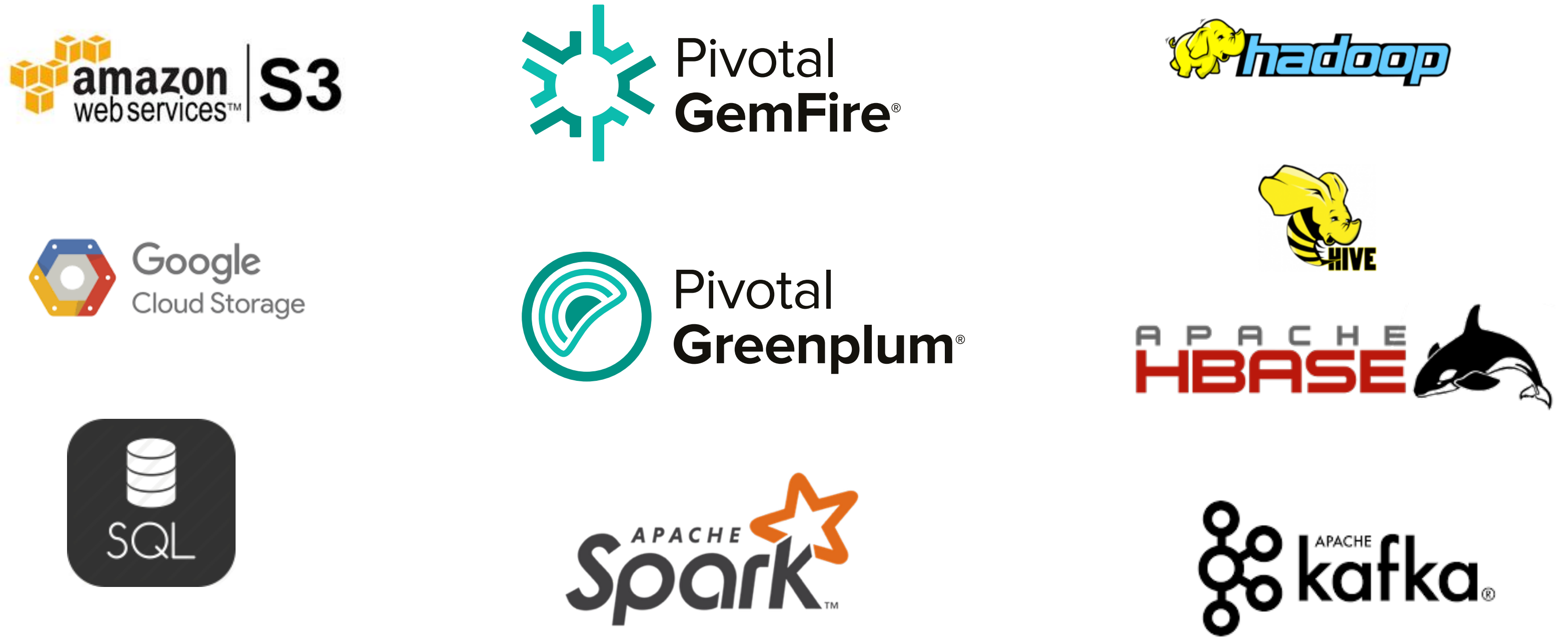
structured data

unstructured data

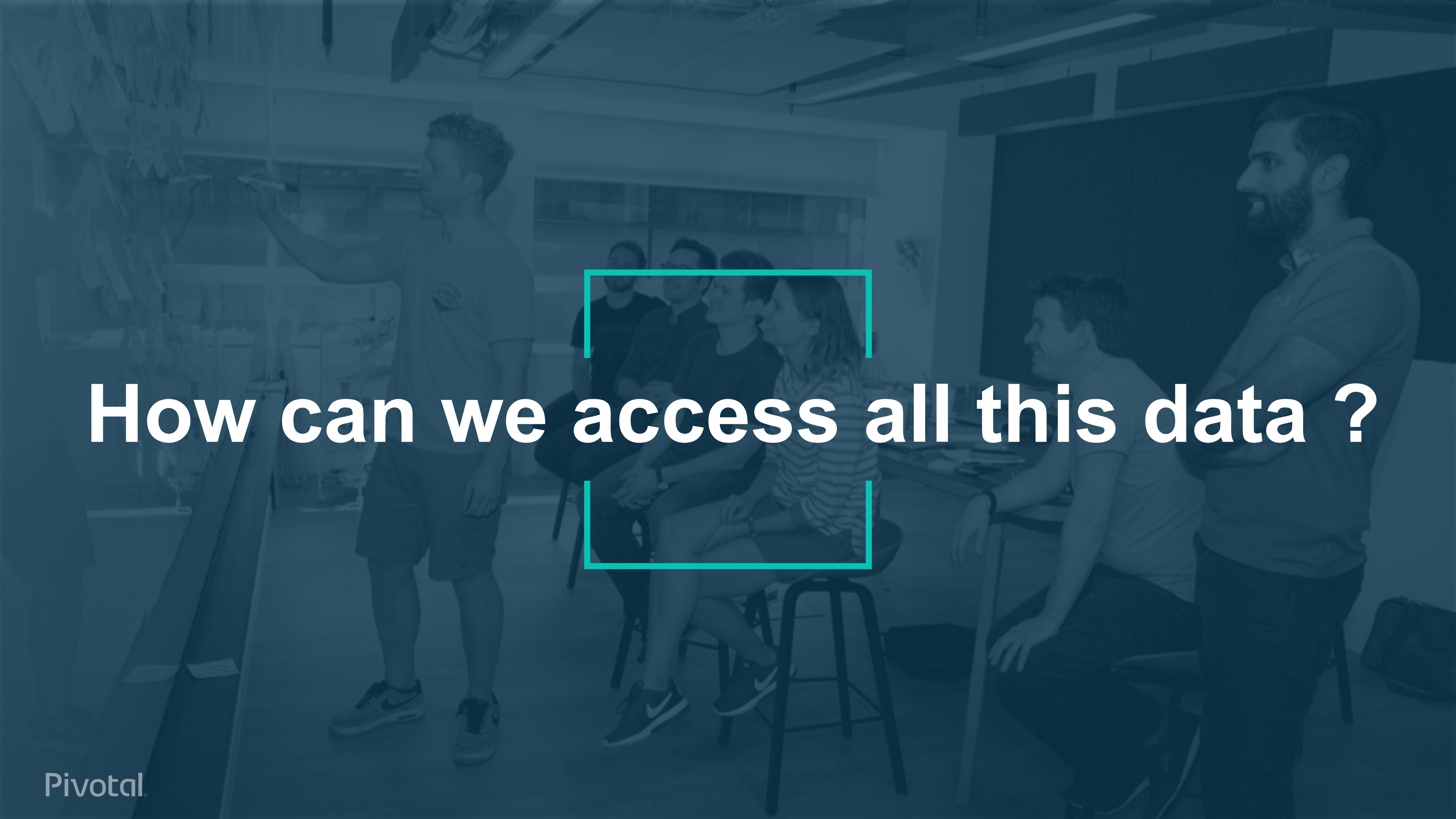
{ semi-structured data }

raw data

Modern Enterprise : wide variety of data engines



RDBMS



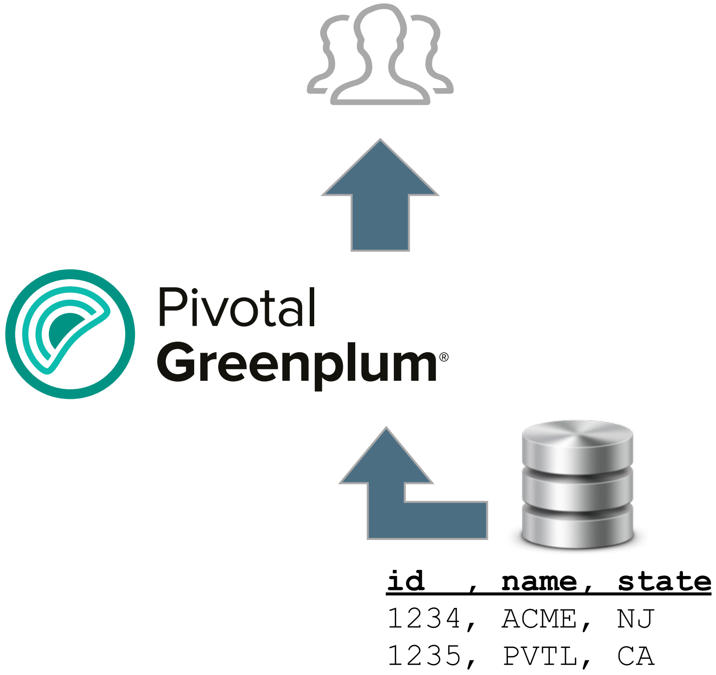
How can we access all this data ?

Managing internal data

Customer information is stored in native Greenplum tables

Find all customer names in CA:

```
SELECT c.name
FROM   customers c
WHERE  c.state = 'CA'
```



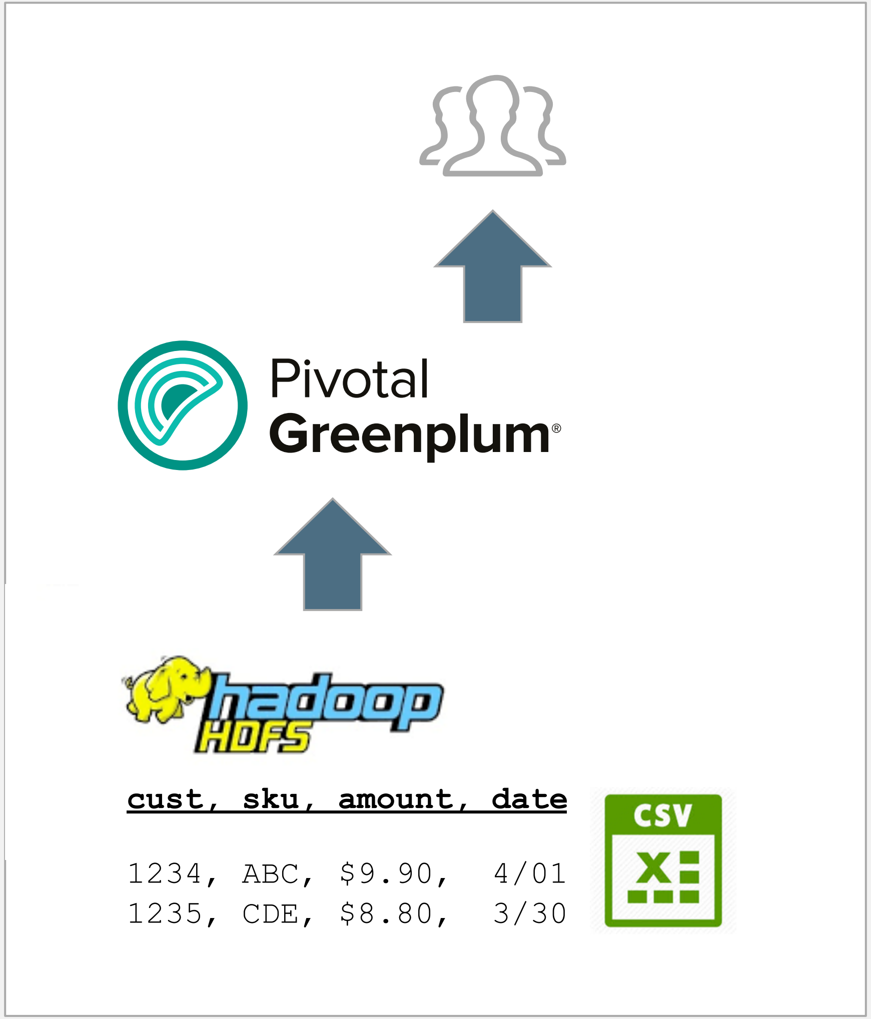
```
id  , name, state
1234, ACME, NJ
1235, PVTL, CA
```

Viewing external data

Order transactions are stored as CSV files in HDFS

Find all orders from today:

```
SELECT *
FROM    orders o
WHERE   o.date = NOW()
```



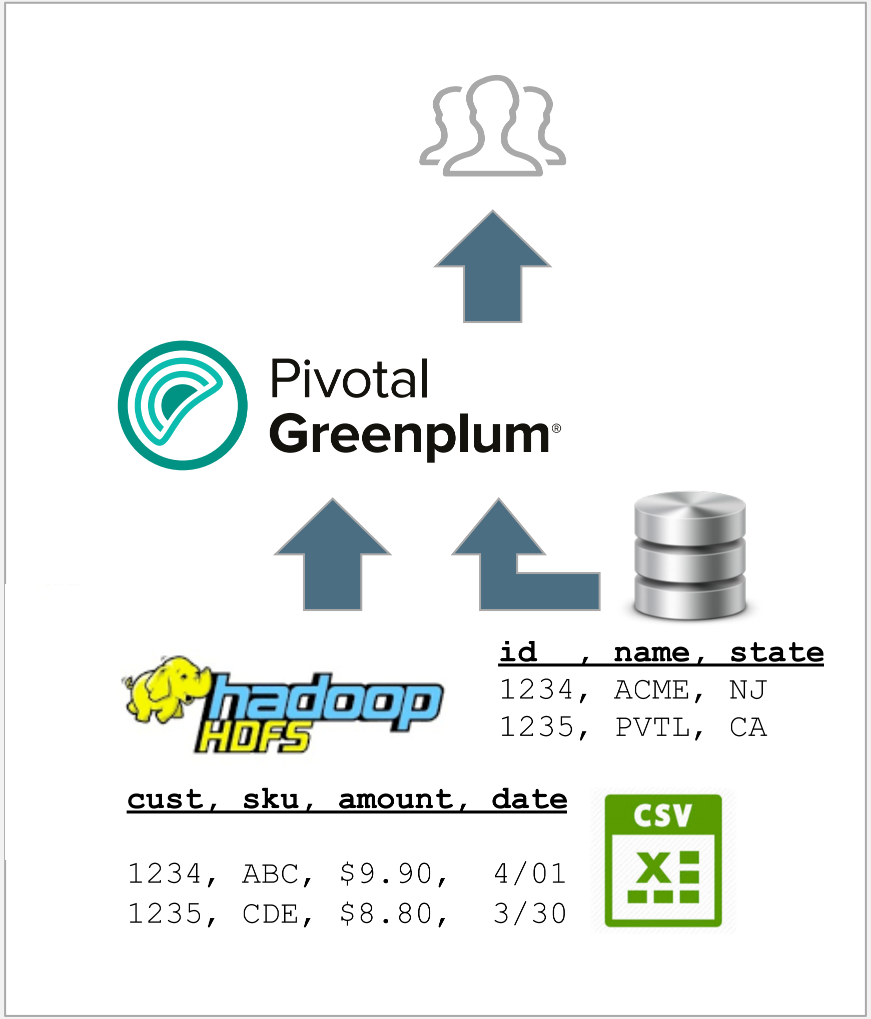
```
cust, sku, amount, date

1234, ABC, $9.90,  4/01
1235, CDE, $8.80,  3/30
```

Joining with external data

Merge order and customer data from different data sources

Find all orders from today, including customer names:

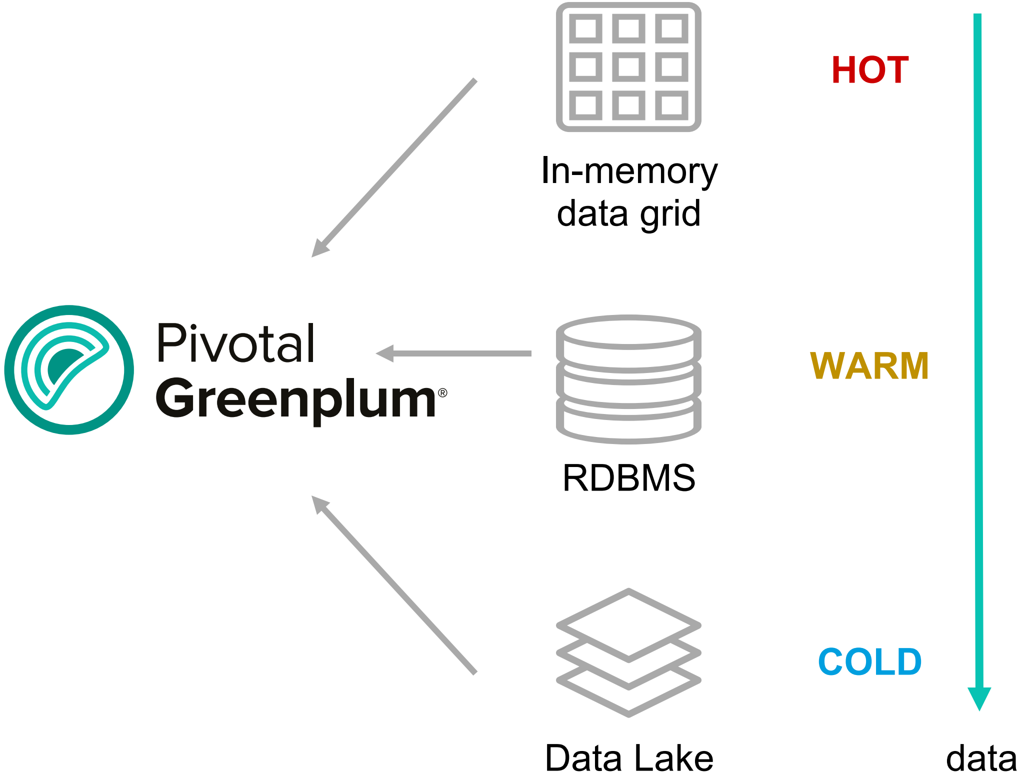
```
SELECT c.name, o.amount
FROM   customer c, sales s
WHERE  s.date = NOW()
AND    c.id   = s.cust
```



```
id    , name, state
1234, ACME, NJ
1235, PVTL, CA
```

```
cust, sku, amount, date

1234, ABC, $9.90,  4/01
1235, CDE, $8.80,  3/30
```

Analytics across data of wide time range



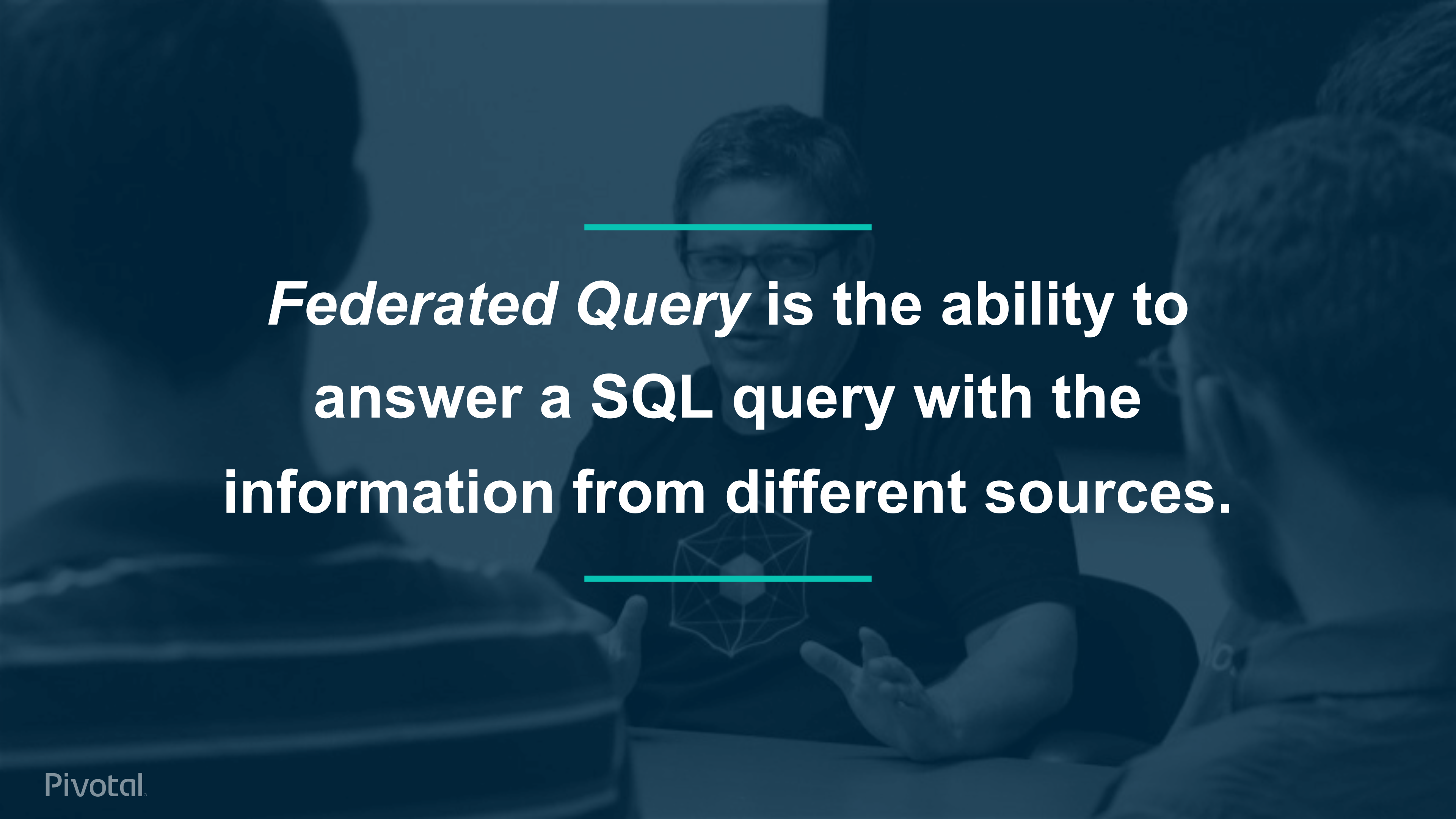
Data is stored in different systems based on operational requirements

Can I work with data created *5 seconds* ago ?

Can I run a report on data from *5 months* ago ?

Can I inspect the data archived *5 years* ago ?

Data is available for analytics with Greenplum no matter where it resides !



Federated Query **is the ability to answer a SQL query with the information from different sources.**

Greenplum External Table

Provides the definitions for:

- the ***schema*** of the external data
- the ***protocol*** used to access the data
- the ***location*** of the data in an external system
- the ***format*** of the external data

Participates in query execution and allows plug-in connectors to external data for different protocols.

```
CREATE [READABLE] EXTERNAL TABLE table_name
( col_name data_type [,...] | LIKE other_table )
LOCATION ('<protocol>://<path to data>...)
FORMAT 'TEXT'
```

```
CREATE WRITABLE EXTERNAL TABLE table_name
( col_name data_type [,...] | LIKE other_table )
LOCATION ('<protocol>://<path to data>...)
FORMAT 'CUSTOM'
  (Formatter=<formatter_specifications>)
[ ENCODING 'encoding' ]
```

```
CREATE [READABLE] EXTERNAL WEB TABLE
table_name ...

CREATE WRITABLE EXTERNAL WEB TABLE table_name
...
```

External Protocol

- Provides ***connectivity*** to an external system
- Implements methods to ***read data*** from the external system and ***write data*** into it
- Defines the ***validation logic*** for external table specifications
- Can be packaged as a ***shared library*** file (`.so`) and loaded dynamically

```
                AVAILABLE PROTOCOLS


file://     -- for files on Greenplum segments

gpfdist:// -- for files on remote hosts

s3://       -- for files in AWS S3 bucket

gphdfs://   -- for files in Hadoop HDFS

http://     -- for WEB tables


pxf://      -- for data sources with JAVA APIs :

             ●   files in Hadoop HDFS
             ●   data in Apache Hive  tables
             ●   data in Apache HBase tables
             ●   rows in RDBMS tables via JDBC
             ●   objects in in-memory grids
             ●   messages in queues
             ●   ... build your own adapter ...
```

Platform Extension Framework (PXF)

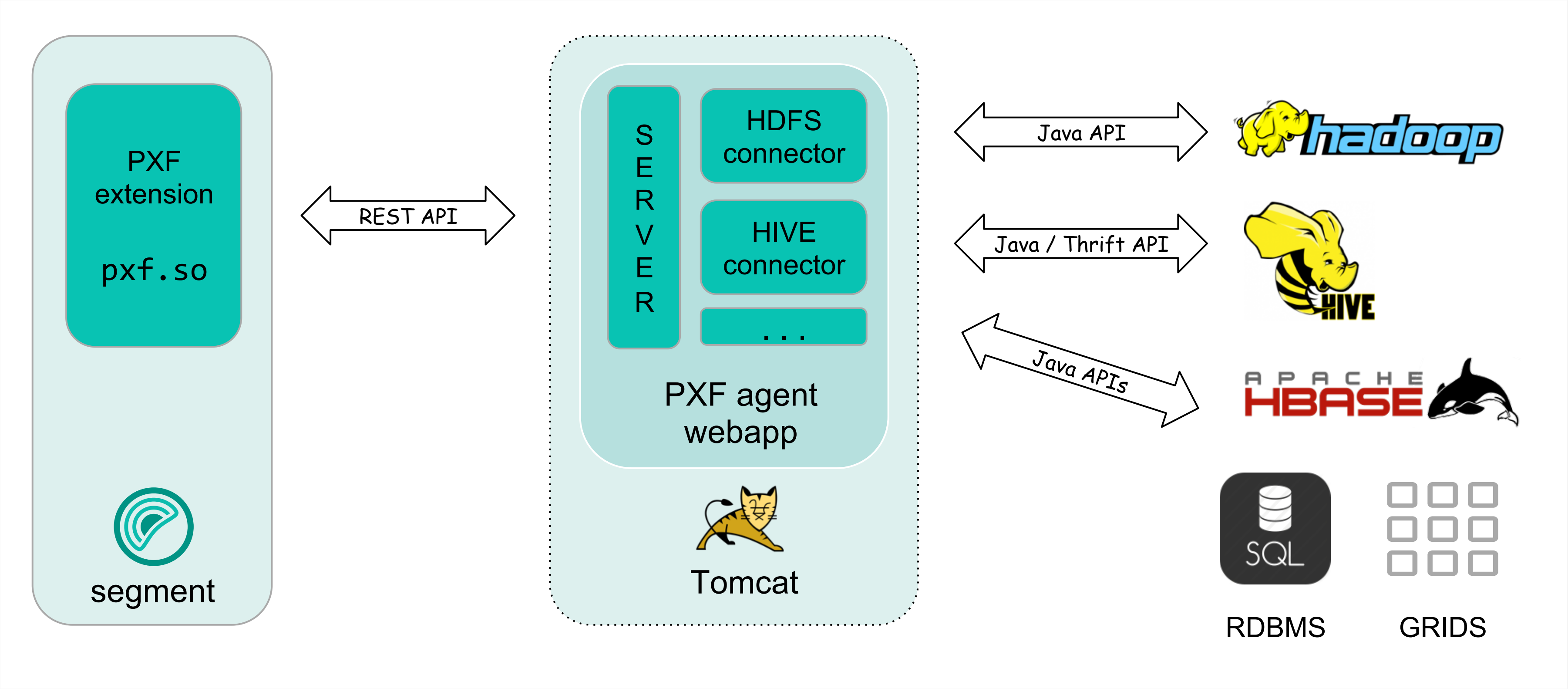
The Platform Extension Framework (PXF) provides:

- parallel, high throughput data access
- federated queries across heterogeneous data sources
- built-in connectors that map a Greenplum Database external table definition to an external data source.

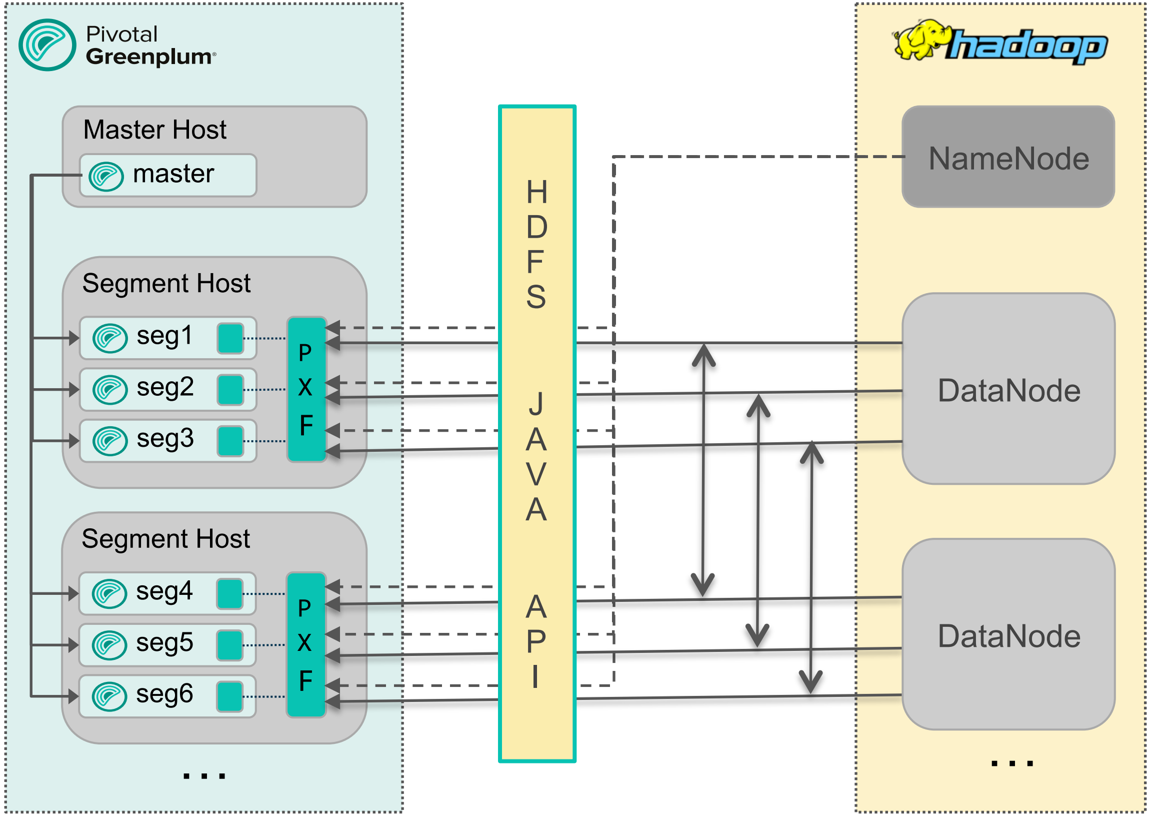
Available in Pivotal Greenplum® since 2017 (5.1 release)

- **PXF is originally a part of Apache HAWQ (incubating) launched in 2012 and open-sourced in 2015**
- **PXF is used to connect to data in Hadoop ecosystem**
- **PXF is open-sourced under the Apache license**

PXF > Architecture



PXF > HDFS Data Import Flow



1. Master submits a query and segments start parallel execution

2. Each segment query execution slice gets a thread in PXF JVM

3. PXF asks HDFS Namenode for the information on file fragments

4. PXF decides on a workload distribution among threads

5. PXF reads data fragments via HDFS APIs from Datanodes and passes it to segments

6. Segments convert data into tuples and return them to Master

PXF Fragmenter

Functional interface which

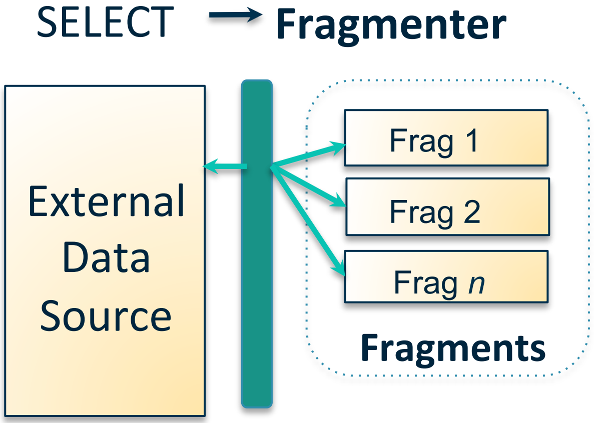
splits data from an external data source

into a **list of independent fragments**

that can be read in parallel.

Examples of a fragment:

- FileSplit in HDFS
- Table partition in JDBC



PXF Accessor

Functional interface which

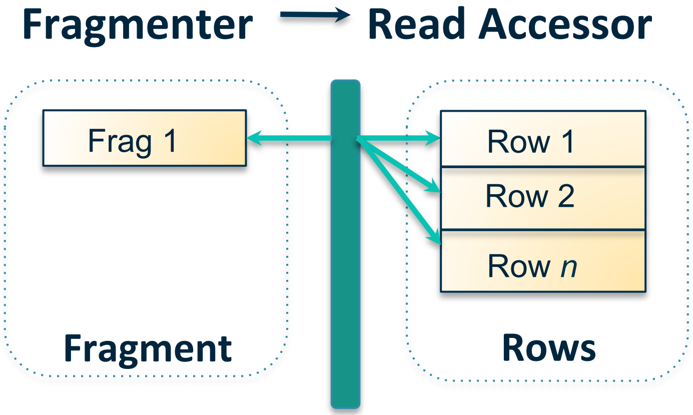
reads a **single fragment**

from an external data source and

produces a **list of records/rows**.

Examples of a record:

- Line in a text file
- Row in a JDBC ResultSet



PXF Resolver

Functional interface which

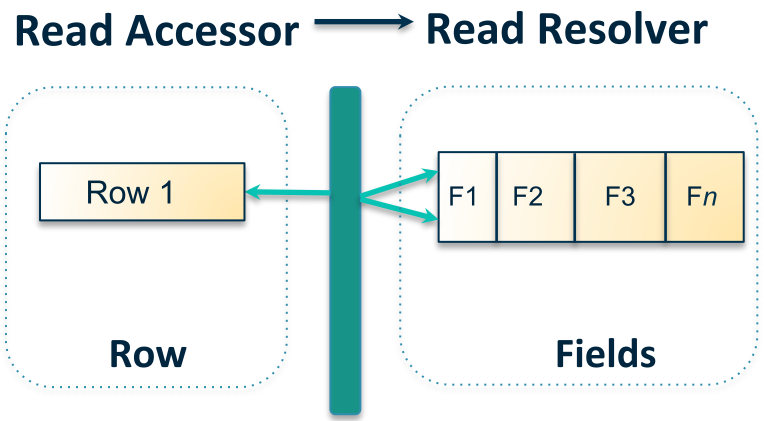
deserializes a record/row into **fields** and

transforms the data types

into those supported by Greenplum

Examples of a field:

- Value between commas in a CSV line
- Column value in a JDBC ResultSet



PXF Profile

A profile is a simple **name mapping** to

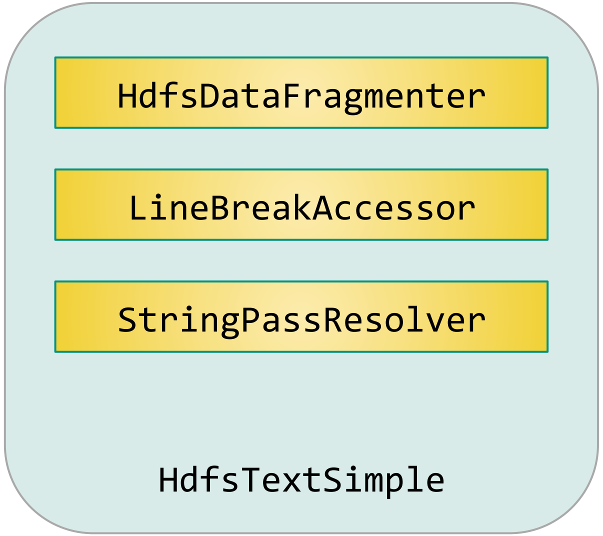
a set of connector plug-in class names

implementing

Fragmenter, Accessor and Resolver

functional interfaces.

Profiles are useful when defining PXF external tables in Greenplum



PXF External Table

Register PXF Greenplum extension

Define an external table with:

- the schema that corresponds to the structure of external data
- the protocol `pxf://` and the location of the data on external system
- the profile to use for accessing the data
- the format of data returned by PXF

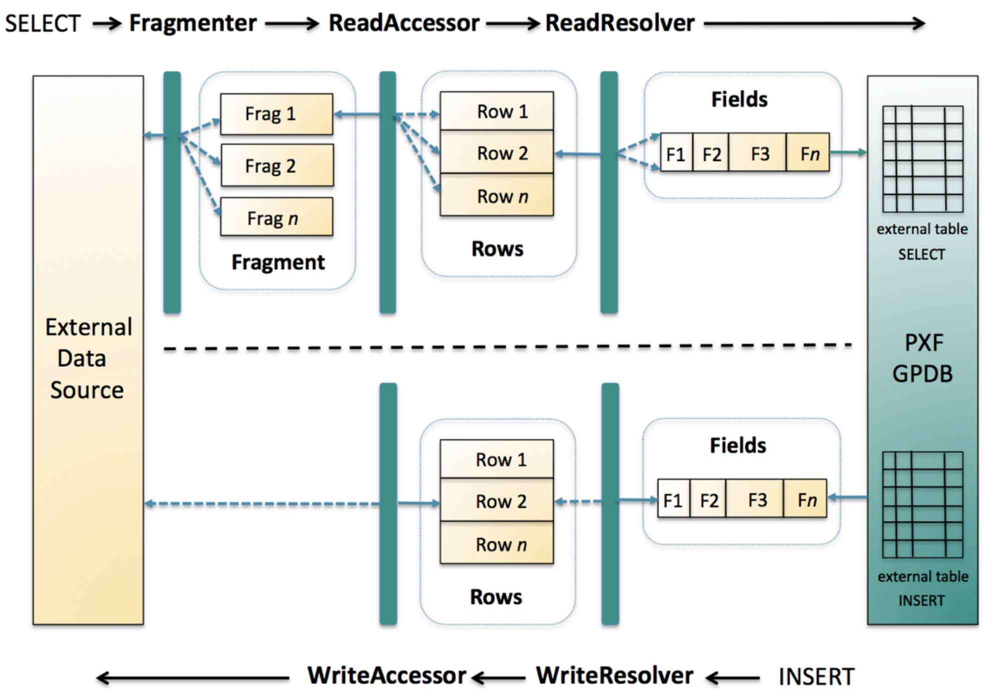
hadoop HDFS

```
cust, sku, amount, date

1234, ABC, $9.90,  4/01
1235, CDE, $8.80,  3/30
```

CSV

```
-- create extension only once per database
CREATE EXTENSION pxf;

-- define external table

CREATE EXTERNAL TABLE sales

(cust int, sku text, amount decimal, date date)

LOCATION

('pxf:///2018/sales.csv?PROFILE=HdfsTextSimple')

FORMAT 'TEXT'
```

PXF > Data Flows Summary



Fragmenter, Accessor and Resolver are working in combination to process data

They can be specified as a pre-built profile or independently

Greenplum external table defines data schema, location, format and the profile to use to get the data

PXF can read the data from the external system or write to it

PXF > HDFS Connector

Data Format	Profile Name	Description
Text	`HdfsTextSimple` `HdfsTextMulti`	Read delimited single or multi-line records from plain text data on HDFS.
Parquet	`Parquet`	Read Parquet format data (`<filename>.parq`).
Avro	`Avro`	Read Avro format binary data (`<filename>.avro`).
JSON	`JSON`	Read JSON format data (`<filename>.json`).

PXF > Hive Connector

File Format	Profile Name	Description
TextFile	`Hive, HiveText`	Flat file with data in comma-, tab-, or space-separated value format or JSON notation.
SequenceFile	`Hive`	Flat file consisting of binary key/value pairs.
RCFile	`Hive, HiveRC`	Record columnar data consisting of binary key/value pairs; high row compression rate.

ORC	`Hive, HiveORC,`
	`HiveVectorizedORC`
		Optimized row columnar data with stripe, footer, and postscript sections; reduces data size.

Parquet	`Hive`	Compressed columnar data representation.

PXF > Other Connectors

- Apache HBase connector
- JDBC connector (community)
- Apache Ignite connector (community)
- Alluxio connector (community)

Advanced Topics > Data Processing Optimizations

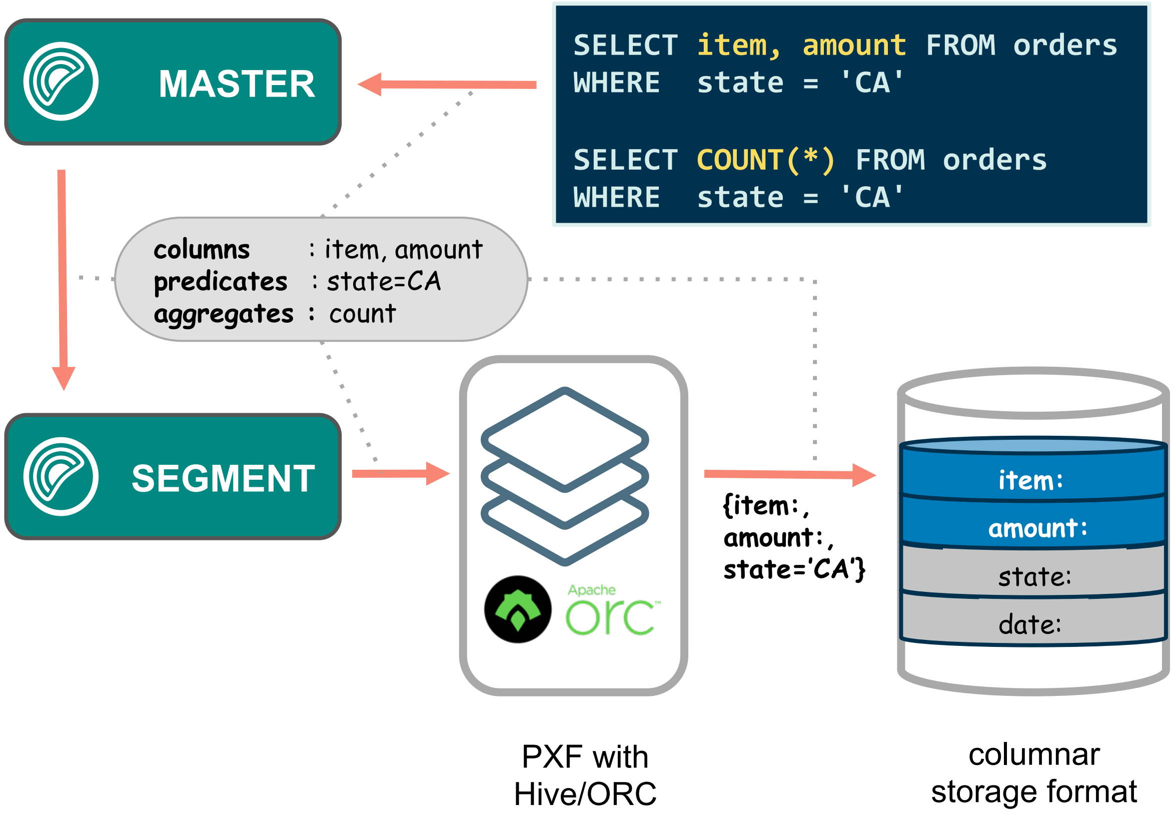
- Avoid data deserialization -- read chunks of text and stream to Greenplum without “resolving” in PXF

- Columnar vectorization -- resolve all row values for a given column at once

- Send multiple rows in batches

- Limit amount of data read from an external system and sent over the network

Advanced Topics > Column Projection



```
SELECT item, amount FROM orders
WHERE  state = 'CA'

SELECT COUNT(*) FROM orders
WHERE  state = 'CA'
```

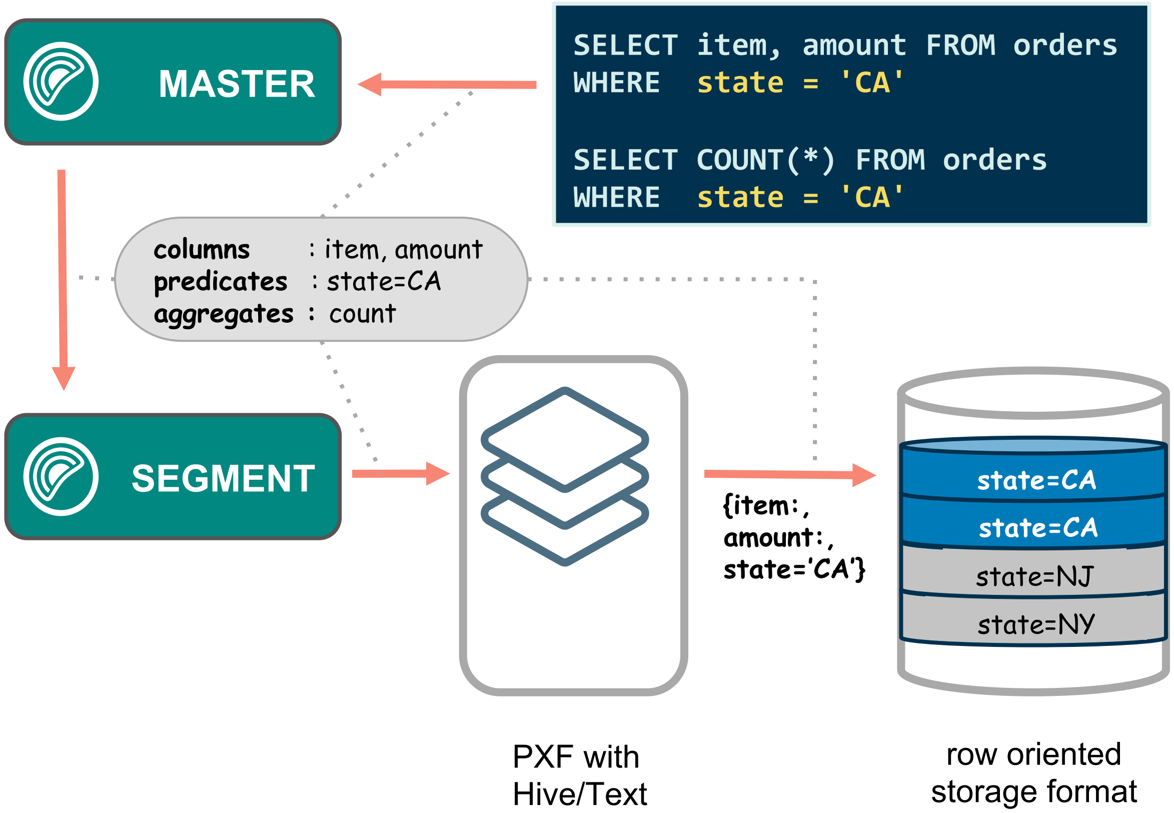
Pushing information about requested columns all the way down to the external system improves performance

Avoids sending unnecessary columns over the network from PXF to Greenplum

Avoids reading unnecessary columns from the disk

Similar benefits can be obtained for some aggregate queries

Advanced Topics > Predicate Pushdown



```
SELECT item, amount FROM orders
WHERE  state = 'CA'

SELECT COUNT(*) FROM orders
WHERE  state = 'CA'
```

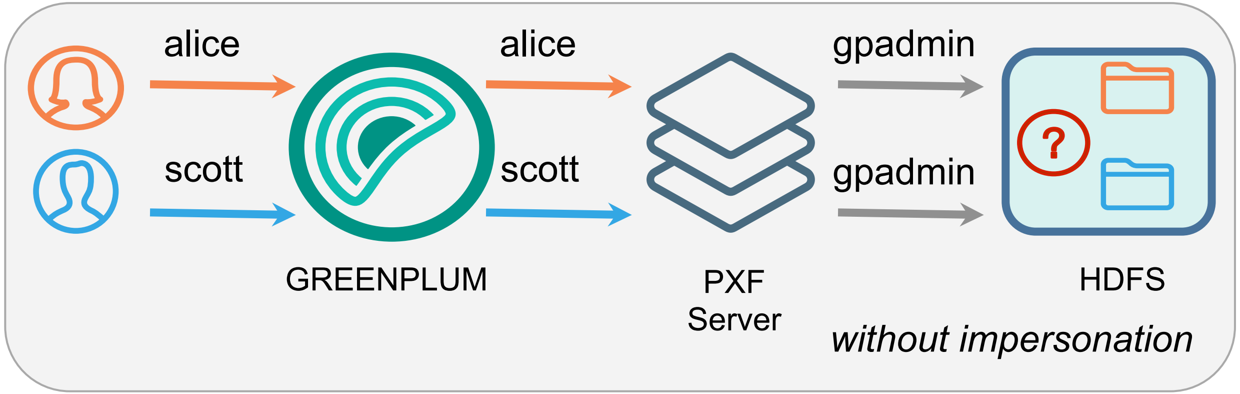
Pushing information about filter conditions (predicates) all the way down to the external system improves performance

PXF itself does not evaluate predicates

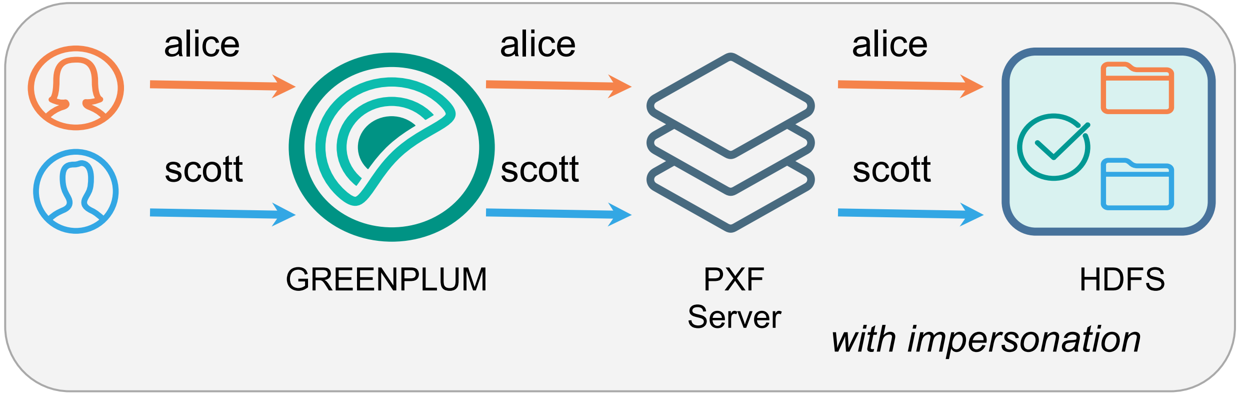
But external system might support predicates for its own queries (e.g. JDBC)

A predicate might cause the whole partition to be eliminated from consideration (e.g. Hive)

Advanced Topics > User Impersonation



without impersonation



with impersonation

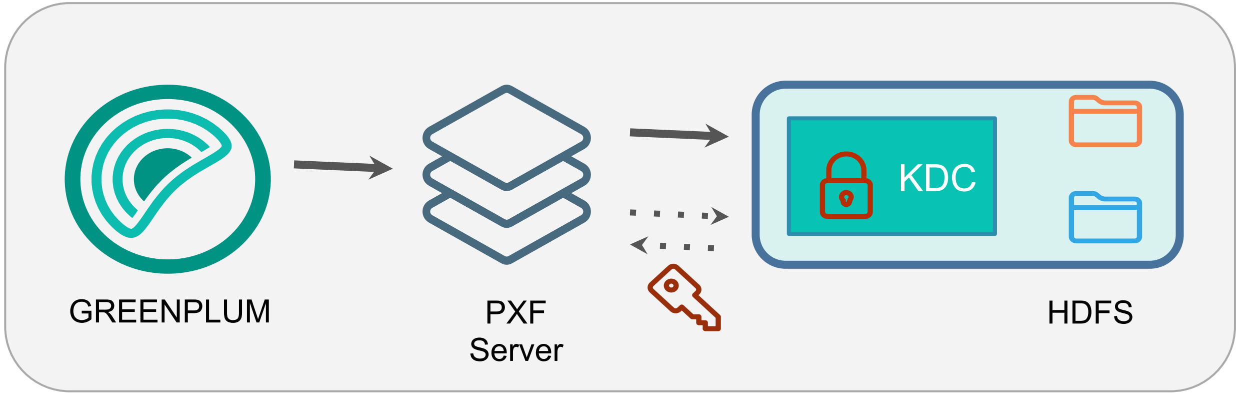
Allows the PXF server to submit requests to external systems on behalf of Greenplum end-users

Must be explicitly supported by the PXF connectors

Prevents the need to grant the PXF server OS user 'gpadmin' superuser access in the external system

Allows to preserve fine-grained access control setting in the external system

Advanced Topics > Kerberos Security



A Hadoop cluster secured with Kerberos requires strong authentication to services based on keys and tickets

The PXF server registers service principal with the Kerberos KDC and stores its secret in a keytab file on a local file system

The PXF server uses the key in the keytab file to obtain a ticket to access resources in Hadoop cluster, such as files in HDFS

Summary

You can contact me at:

Alexander Denissov

✉ adenissov@pivotal.io

in www.linkedin.com/in/denissov

- ➢ Reviewed the Federated Query concept
- ➢ Explored Greenplum External Tables
- ➢ Learned about PXF and its architecture
- ➢ Understood how to use Greenplum with PXF for creating federated queries across multiple data sources, data engines and data formats

More information at:

https://greenplum.org
https://github.com/greenplum-db/gpdb
https://github.com/apache/incubator-hawq/tree/master/pxf
http://gpdb.docs.pivotal.io/570/pxf/overview_pxf.html

Pivotal®

Transforming How The World Builds Software

© Copyright 2017 Pivotal Software, Inc. All rights Reserved.