# Chemistry Knowledge Handling: With GemFire and Spring DataFlow

JUNE 13, 2018 [GregorY GREEN](https://content.pivotal.io/authors/dan-baskette)

|  |  |
| --- | --- |
|  | There are trends in large pharmaceutical companies to buy commercial chemistry knowledge management products rather than build in-house applications. Different departments sponsor different solution based on needs. |

Often there is not an aggregated view of chemistry knowledge. For example, one product manages a catalog of chemicals, one holds inventory information that determines the availability of these products and another manages project experiments. In these environments, Scientist sometimes struggle to answering the questions;

* What chemicals have been made?
* How can I get the chemicals I need for my experiments?
* What can be learned about a given chemical’s characteristic?

Most companies find it hard to integrate decades of legacy chemistry data. There is a lack of a centralized data view. Traditional data warehouses are difficult to keep up to date. There is a high cost for high-performance computing infrastructure where improved speed is needed. Dataset are only getting bigger. The typical storage mechanisms are unable to scale.

The key challenges areas are complexity, scalability, availability and support.



### Complexity

There are increasing performance challenges as the data grows. More products require data integration. Real-time analytical analysis across multiple areas could impact critical chemistry applications. Data marts are warehouses are often views of stale data.

### Scalability

There may be limited scalability support issues in existing chemistry management data systems. For example, database cartridges allow scientist to perform expensive molecule comparisons operations. Larger data sets require very expensive shared hardware (big boxes) to meet performance needs across multiple applications.

### Availability

The lack of scalability often results in a single point of failure for products. This is high risks to company’s productivity for critical chemistry information products.

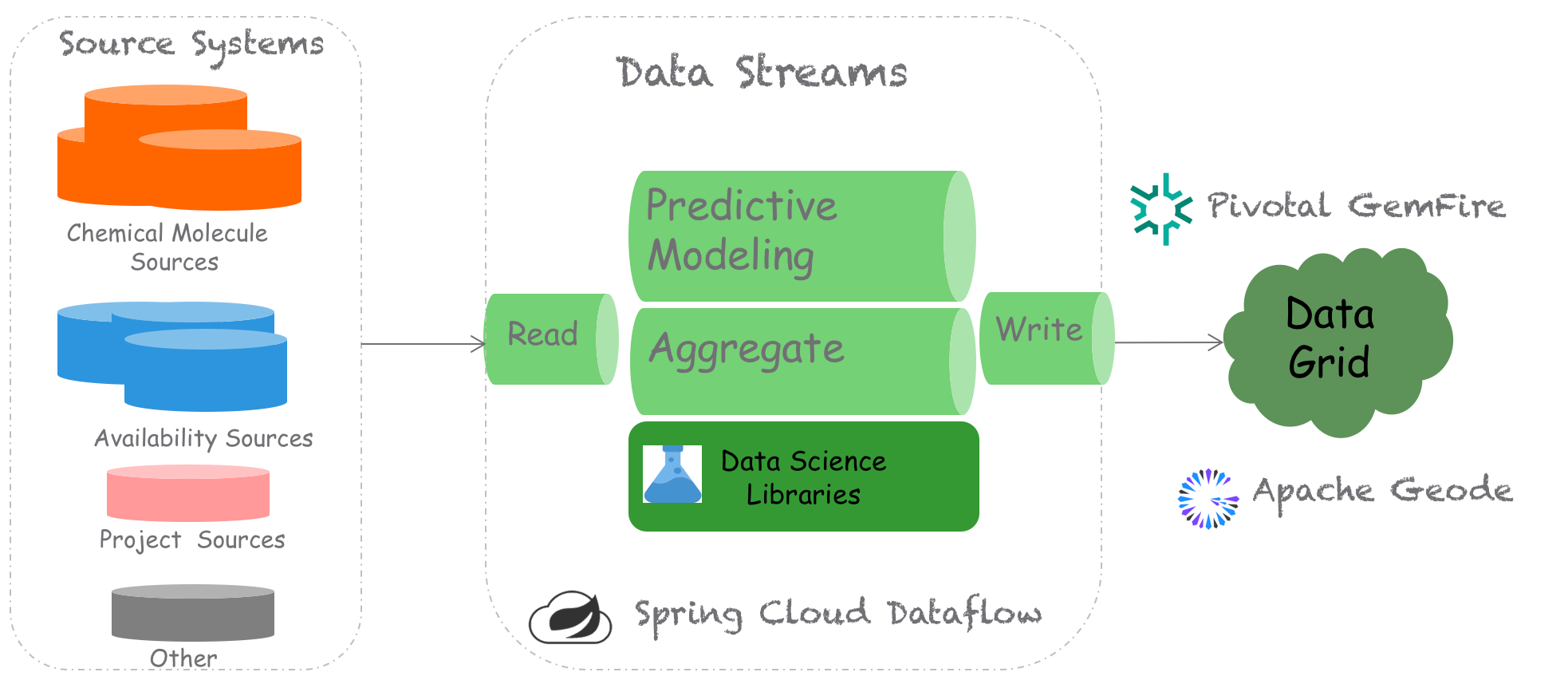
### Support

Efforts to create specialized/replicated data marts/warehouses to assist specific application data processing needs can raise maintenance and information management concerns.

## Conceptual Architecture

Data integration and caching platforms like Spring Cloud DataFlow (SCDF) and Pivotal GemFire that can be help with some of these challenges. In this scenario, SCDF can be used to stream data from source commercial chemistry products to Pivotal GemFire cache based on configurable flows.

The following is a logical view of a hypothetical architecture.



## Spring DataFlow

Spring Cloud Data Flow (SCDF) is data integration platform that supports real-time data processing pipelines. This allows users to deploy long or short lived data processes respectively.

SCDF supports a pipeline definitions language that is very similar to UNIX command. In the following example, in UNIX the file piped the output of the ”ls” into “wc” to counts the words.

|  |
| --- |
| ls | wc |

Similarly, in SCDF you can chain out of box and custom modules to form very flexible processing pipelines.

The following hypothetically flows start with a file source. It then sends the content to a custom processor to build a molecule object. It then sends the SCDF output to a custom process to build the molecule information. The last module (called a sink) saves the output Molecule objects to GemFire.

|  |
| --- |
| file | molecule | gemfire |

SCDF supports a REST API to allows users to definite, start and monitor flows. There is shell interface exposes the management of the flows thru an interactive command line prompt. SCDF also contains a GUI dashboard managing and monitoring data pipelines

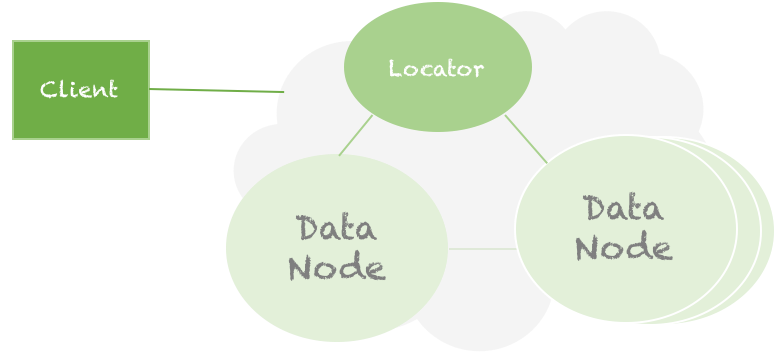
Each pipeline in the data flows consist of [Spring Boot](https://projects.spring.io/spring-boot/) apps. using the [Spring Cloud Stream](https://cloud.spring.io/spring-cloud-stream/) or [Spring Cloud Task](https://cloud.spring.io/spring-cloud-task/) (microservice frameworks. Spring Cloud is used to long lived flows and Spring Cloud Task is used to short-lived process flows. Spring Boot can also wrap calls to scripting languages (like python, ruby, groovy, etc.) or other REST services. The flows can be defined and maintained by Scientists and or administrators.

## GemFire

GemFire is an In-Memory Data Grid (IMDG) that is powered by the Apache Geode open source project. GemFire can be used as a data cache for outputted data flows.

### How does it work?

GemFire's two main components are a locator and cache server (a.k.a. data node).



The client can connect to a locator to access its data. All data is stored on the cache server. Cache servers/data nodes register with locators to be discovered by clients or other data nodes. The knowledge of data nodes and where data is located is abstracted from the client. The number of data nodes can be scaled up to handle increased data or clients.

### GemFire Regions

Data is managed in a [region](http://gemfire.docs.pivotal.io/93/geode/basic_config/data_regions/chapter_overview.html). It is similar to a table in a traditional relational database. Each region can have a different data policy. A replicated region data policy stores a copy of entries on each data node. Data is split in a partition region data policy so that each data node only stores pieces of the primary values in entries and or a configured number of backup copies.

**Data Access**

GemFire supports NO SQL operations to get region entry objects very quickly by a key. Regions are based on key/value pairs.

|  |
| --- |
| Region<String, Molecule> molecules |

Users can store entry in a region using a "put" operation.

|  |
| --- |
| molecules.put(key,molecule) |

Users can retrieve an entry object directly from a region "get" operation by providing the key identifier. See the coding example below.

|  |
| --- |
| Molecule molecule = molecules.get(key) |

It supports SQL like queries through its Object Query Language (OQL). With [QOL](http://gemfire.docs.pivotal.io/93/geode/developing/querying_basics/chapter_overview.html) you can select objects by a particular attribute in a where clause.

For a "molecules" region, QOL would allow you to perform a search such as

|  |
| --- |
| select \* from /molecules where smile = 'C' |

|  |
| --- |
| select \* from /molecules where formula = 'H20' or (weight > 18.0 and weight < 20) |

It supports simple and complex queries (nested queries).

**Other Features**

GemFire supports other features such as events listeners (similar to database triggers), joins, transactions, functions (similar to stored procedures), full-text searches and more. The GemFire use case tends to be based on FAST data access patterns. See <https://pivotal.io/pivotal-gemfire> for more details on GemFire.

## Demo Implementation

**Pivotal Cloud Foundry (PCF)**

The demo implementation of this application is deployed on Pivotal implement of Cloud Foundry, named Pivotal Cloud Foundry. Cloud Foundry is an opinionated cloud-native platform that forces implement to maintain the 12-factor based principals. See <https://12factor.net/> for details.

**Pivotal Cloud Cache (PCC)**

PCF allows users to manually or auto scale application instances to meet increase capacity demands. Cloud native applications are stateless. Any information that is stateful must be available to all application instances. Cached state information can be managed in a backing service. GemFire implementation of a caching backing service is called [Pivotal Cloud Cache](https://docs.pivotal.io/p-cloud-cache/1-0/index.html) (PCC).

PCC allows the user to quickly provision multiple environment instances of GemFire. An instance can be based on a desired plan that dictates the number of locators/data nodes and the size of the cache. Complex environments can be quickly created with simple commands

Example command to create a PCC instance

|  |
| --- |
| cf create-service p-cloudcache <plan> <name> |

You can bind PCC service instances to application instances so that the connections and credentials details are securely provided to consuming applications.

|  |
| --- |
| cf bind-service <app> <name> |