**Utilizing multiple data stores and data models: Is Polyglot Persistence worth It?**

Polyglot Persistence is a process for storing data in the best database available, no matter the data model and data storage technology. This process is based on the understanding different data stores will handle certain types of data better than others. Polyglot Persistence is, unfortunately, not available for easy “downloading”, but must be designed for the unique Data Architecture of each individual enterprise. This “storage philosophy” is a recent development, and still needs to evolve, technologically.

The words “poly” and “glot” are Greek in origin, but were assembled for use in English during the 17th century. The word polyglot means to speak and write in multiple languages. The term “Polyglot Programming” appeared in 2006, to describe the understanding certain computer languages are excellent at solving specific problems, while others are not, and for this reason, programs should include multiple languages. The persistence part of Polyglot Persistence refers to memories that are “saved” in a safe way or location, allowing them to be “persistent.” Polyglot Persistence uses a program that communicates with multiple types of databases, and uses the most “appropriate database” to store and process data.

Polyglot Persistence provides support for multiple database models and is able to utilize the best data model for the job. The Polyglot Persistence philosophy comes with its own “technical” strengths and weaknesses. Polyglot Persistence provides some key benefits:

* Simplifies Operations: Different databases, coordinating and working with one another, make for complicated operations and cause fragmentation. Polyglot Persistence simplifies operations and helps to select the best component for the situation, helping to eliminate fragmentation.
* Faster Response Time: All the features of databases in the program are leveraged, improving response times.
* Efficiency: The ElasticSearch app can return results using “relevance” as a priority in the listings, while MongoDB cannot. A Polyglot Persistence application would “automatically” assign relevance-oriented processing.

Utilizing a Polyglot Persistence model can be both difficult and expensive. Specialists often need to be brought in to integrate the different databases. These expenses should be given serious consideration in evaluating long term goals. Other problems to consider are:

* + Permanent IT staff will need training on the “new” systems.
  + Maintenance and repairs can be time consuming, because running tests is difficult. If data is sharded into many databases, the testing of data layers can become complicated. Debugging is, of course, also quite time consuming.
  + Making sure a system with multiple components is fault-tolerant is difficult, to say the least. Integrating multiple databases requires a significant amount of operational and engineering expenses. A business will need to have experts for each database technology. For the program to remain operational, all of the databases must be up and running. This situation makes the fault tolerance the weakest link.

When using two, or more, different kinds of databases, where to store the data becomes a decision. A bad decision could result in having to rework the system, including the time needed to migrate data from one database to another. The person, or team, writing the program should be able to provide useful advice.

Applications also have to deal with the increasing complexity of your Data Architecture, having to communicate with two potentially very different data stores. When done correctly, it should be possible to separate the “persistence layer” (a group of communication files) of the application, and free up the rest of the application to do its work. The more data stores available, the greater the potential for increasing the complexity of the data persistence layer.

**Utilizing multiple data stores and data models: Is Polyglot Persistence worth It?**

Polyglot Persistence is a process for storing data in the best database available, no matter the data model and data storage technology. This process is based on the understanding different data stores will handle certain types of data better than others. Polyglot Persistence is, unfortunately, not available for easy “downloading”, but must be designed for the unique Data Architecture of each individual enterprise.

The words “poly” and “glot” are Greek in origin, but were assembled for use in English during the 17th century. The word polyglot means to speak and write in multiple languages. The term “Polyglot Programming” appeared in 2006, to describe the understanding certain computer languages are excellent at solving specific problems, while others are not, and for this reason, programs should include multiple languages. The persistence part of Polyglot Persistence refers to memories that are “saved” in a safe way or location, allowing them to be “persistent.” Polyglot Persistence uses a program that communicates with multiple types of databases, and uses the most “appropriate database” to store and process data.

Polyglot Persistence provides support for multiple database models and is able to utilize the best data model for the job. The Polyglot Persistence philosophy comes with its own “technical” strengths and weaknesses. Polyglot Persistence provides some key benefits:

* + - Simplifies Operations: Different databases, coordinating and working with one another, make for complicated operations and cause fragmentation. Polyglot Persistence simplifies operations and helps to select the best component for the situation.
    - Faster Response Time: All the features of databases in the program are leveraged, improving response times.
    - Efficiency: The ElasticSearch app can return results using “relevance” as a priority in the listings, while MongoDB cannot. A Polyglot Persistence application would “automatically” assign relevance-oriented processing.

Utilizing a Polyglot Persistence model can be both difficult and expensive. Specialists often need to be brought in to integrate the different databases. Other problems to consider are:

* Permanent IT staff will need training on the “new” systems.
* Maintenance and repairs can be time consuming, because running tests is difficult. Debugging is, of course, also quite time consuming.
* Making sure a system with multiple components is fault-tolerant is difficult, to say the least. A business will need to have experts for each database technology.

When using two, or more, different kinds of databases, where to store the data becomes a decision. The person, or team, writing the program should be able to provide useful advice.

Applications also have to deal with the increasing complexity of your Data Architecture, having to communicate with two potentially very different data stores. When done correctly, it should be possible to separate the “persistence layer” of the application, and free up the rest of the application to do its work. The more data stores available, the greater the potential for increasing the complexity of the data persistence layer.