Neal Sakash

Dr. Paul Buhler

DATA 210:02

May 4th, 2015

Polyglot Persistence

Knowledge of product and user data has become a paramount concern for businesses in recent years and the ability to handle the volume, velocity, and variety of big data has placed a strain on traditional databases. The recent development of diversified NoSQL technologies offer varied solutions to the limits of RDBMS, however they come with their own drawbacks which may prevent companies from investing the time and money to fully utilize their features. One argument is to follow the notion of polyglot persistence and weave NoSQL technologies with an already established RDBMS model to best support different data storage needs. The integration of multiple databases could still create a new set of problems in terms of development and operational costs, therefore a counter argument would be to build multiple models within a single scalable database utilizing ACID properties for synchronicity.

Martin Fowler of Thought Works champions the use of polyglot persistence in database design. He finds fault in the tendency to store various types of data in a single database and performance issues caused by doing so. Using the example of an e-commerce business he breaks down the different data types created by customers and how it is not necessarily in the company’s best interest to store it all in a traditional RDBMS. Customer transactions can be stored neatly in a RDBMS, yet purchase histories, and more importantly the purchase history of related customers, are better kept in a graph store. Not only would that increase the performance of company databases, but valuable business insight could be more easily accessible through easier data analysis. Utilizing NoSQL databases in conjunction with a traditional RDBMS allows a company to choose the right type of storage technology to best serve each aspect of the business. Fowler admits there are drawbacks to implementing such technologies on top of maintaining a company’s existing database, such as deciding exact needs and the costs associated with implementation and employee training, but it seems these problems are minimal when looking at the bigger picture. It would be wise for a company of large-scale capacity to adapt and utilize these new technologies as opposed to losing any form of completive advantage to a business rival.

These problems should not be ignored though. With many database choices comes the resulting confusion for developers. Stephen Pimentel with FoundationDB is critical of Fowler’s praise for polyglot persistence and asserts that building multiple NoSQL databases on top of a traditional RDBMS is a data modeling nightmare. He believes this would restrict the functionality of the additional databases and hinder any grains offered through operational costs. Pimentel’s answer to this problem would be to keep data stored on a single platform but add the ability to create multiple data models for specific business needs. Developers would not have to implement completely new databases to customize this modeling and these models could be synchronized throughout the organization through ACID transactions. The potential problem of users accessing stale data created by eventual consistency in NoSQL systems would be eliminated by ACID support without the need to compromise performance and scalability.

In a perfect world businesses would be able to execute new efficient technologies will minimal expense and personnel strain. A company considering the implementation of polyglot persistence must weigh its potential benefits with the cost and reorganization that comes with it. For a startup seeking investment capital they could easily factor this into their budget; however an established company would have to divert more energy and risk to see the project through. Depending on the type of company and products offered, multiple modeling within an establish RDBMS database may be the best course of action to track different data types until NoSQL applications reach a comparable level of maturity.