

Polyglot Persistence Patterns

Abhishek Tiwari 

Citation: *A. Tiwari*, "Polyglot Persistence Patterns", Abhishek Tiwari, 2012.
[doi:10.59350/1fkt7-e7492](https://doi.org/10.59350/1fkt7-e7492)

Published on: August 29, 2012

Polyglot persistence which takes a hybrid approach to persistence is getting a lot of traction these days. Currently I am reading [NoSQL Distilled: A Brief Guide to the Emerging World of Polyglot Persistence](#) by Pramod J. Sadalage and Martin Fowler. Basic idea behind polyglot persistence is to use specialized databases (both NoSQL and Relational) for different purposes within the same web application. One-database-fits-all is no more the de facto choice for web applications. In fact polyglot persistence is about choice - choice to select best data store for a given data type or purpose. Polyglot persistence leverages the strength of multiple data stores.

In fact polyglot persistence is about choice - choice to select best data store for a given data type or purpose.

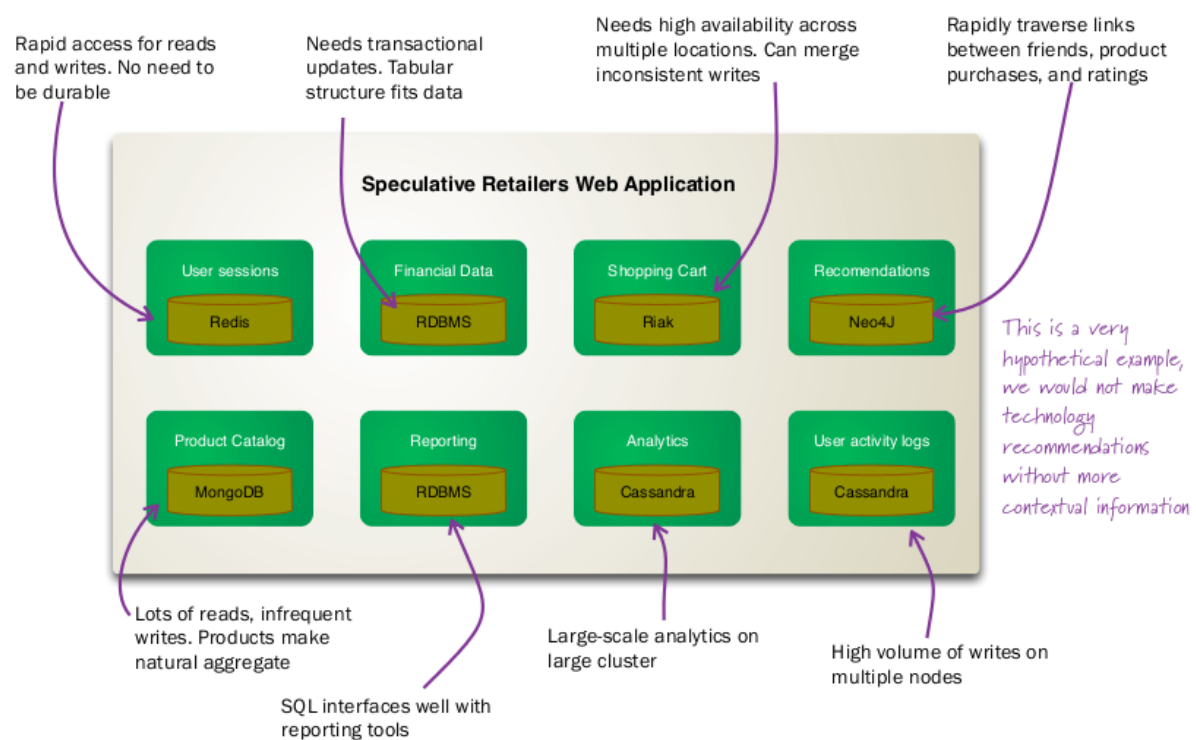


Figure 1: Polyglot persistence pattern for an ecommerce application

Multiple Data Stores

In a polyglot persistence pattern, your web application can use one or more relational databases or combination of relational and NoSQL data stores. Most of modern web frameworks such as Django, Rails etc already support multiple databases. As an example, a polyglot persistence enabled eComerce application can store

- Session data in Redis or Memcached. Session data requires faster read and write but not durability. For better durability we can always use write-through.
- Transactional data (order, payment and account) in a ACID compliant traditional RDBMS stores such as MySQL or Oracle. In addition to that, HBase/Hive with Hadoop can be used to process transaction level data such as order history for market basket analysis.
- Shopping cart data in a high availability and fault-tolerance data store such as Riak or Cassandra. Riak will be a better choice because it is a key/value store with excellent query API with primary key operations such GET, PUT, DELETE, UPDATE.
- Log level data (audit and activity) in a very high write throughput data store such as Cassandra. This is also good for analytic and real-time data mining such as product ranking,
- Data for product recommendations, related products and similar products in a graph database such as Neo4j. Also possible to use Apache Giraph as distributed graph processing infrastructure which runs on Hadoop.
- Product catalogue in a document oriented data store such as CouchDB or MongoDB. Requirement include high read throughput, frequent data change (stock level information) which makes MongoDB a better choice.
- Customer profile data including purchase history, shipping & billing address - again in a document oriented data store. CouchDB is better option because this data change occasionally.

Wrapping Data Store Into A Service

Wrapping data stores into services is another interesting emerging polyglot persistence pattern. Rather than talking directly to data store, application will use wrapped data stores accessible via APIs. For instance MongoDB, Riak, Neo4j all of them provide REST APIs. Once data access is encapsulated into services, underlying data stores can evolve without changing application logic.

Once data access is encapsulated into services, underlying data stores can evolve without changing application logic.

AWS - Polyglot Persistence Solution In The Cloud

Amazon DynamoDB together with S3, RDS, ElastiCache and EMR (MapReduce) provides an interesting and cost effective polyglot persistence solution in the cloud with easy scalability and low administration overhead.

Using AWS as polyglot persistence solution has several advantages. First of all most of AWS service are managed and highly scalable. Second, self-managed multiple data stores altogether can be a big undertaking - simply too expensive and complicated. For self-managed polyglot persistence solutions deployment complexity is very high especially managing multiple environments.

For self-managed polyglot persistence solutions deployment complexity is very high

Deployment simplicity is one of the major selling points for a AWS based polyglot persistence solution - in most cases one click away. Having said, in terms of features AWS services can be equally powerful. For instance, an example AWS polyglot persistence solution for an eCommerce will look like

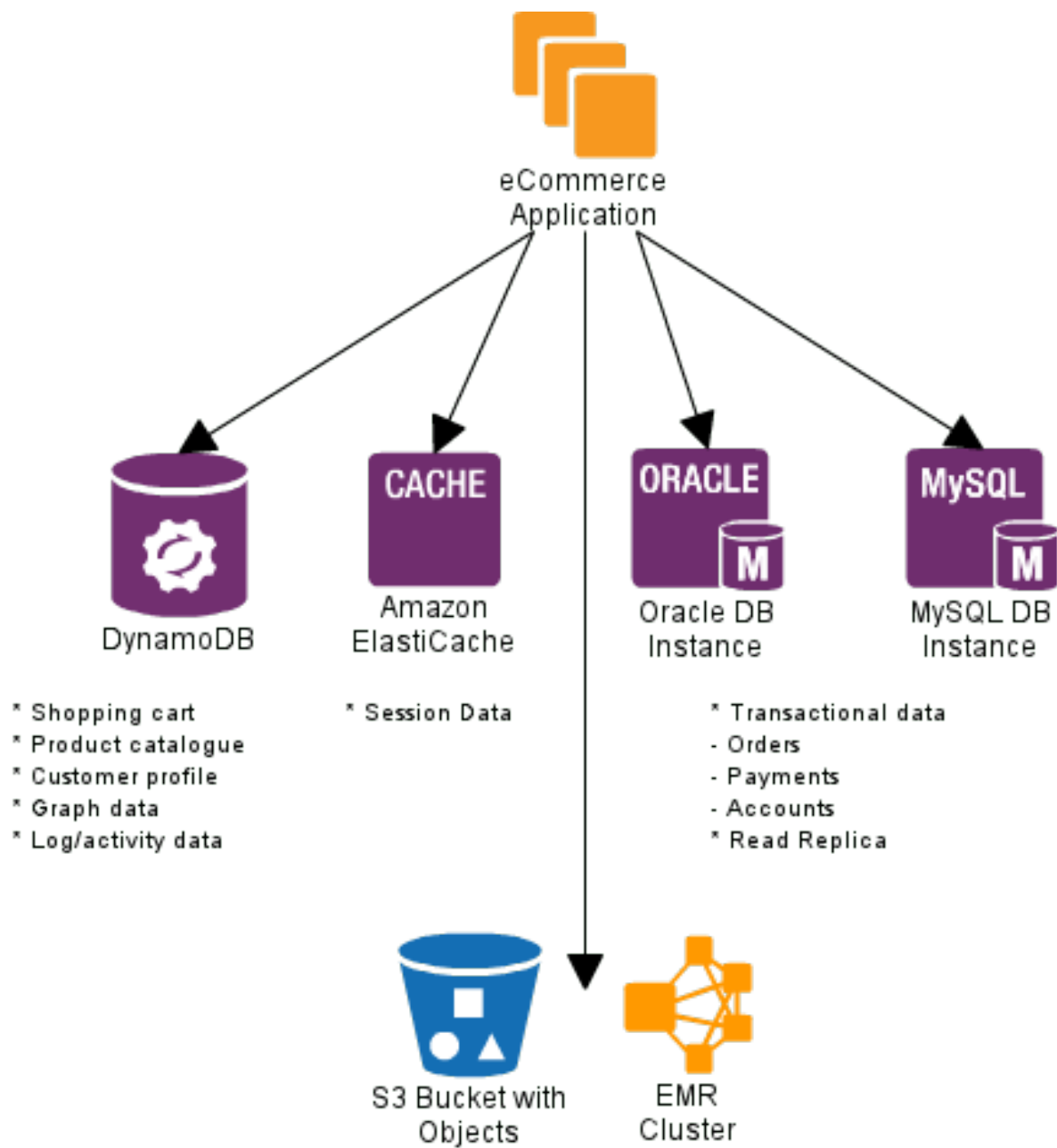


Figure 2: Polyglot persistence patterns in AWS

- ElastiCache (Memcached) for session data.
- DynamoDB for shopping cart, product catalogue, customer profile, graph data, log/activity data.
- RDS (MySQL, Oracle, Sql server) for transactional data. Use read-replica for high read throughput.
- S3/EMR for batch processing of logs and transactional data.