Polyglot Persistence
Boon and Bane for Software Architects

Dipl.-Inf. Univ.
Ralf S. Engelschall
msg Applied Technology Research

msg systems
About Speaker:

Ralf S. Engelschall

• Computer Scientist and Open Source Hacker.
• Founder of Apache Software Foundation (ASF), OpenSSL and OpenPKG.
• Author of popular Open Source solutions, including Apache mod_ssl, Apache mod_rewrite, GNU pth, GNU shtool, etc.
• Principal Software Architect and Department Manager at msg Applied Technology Research, cross-division of msg systems, Munich, Germany.
Nothing, not all the armies of the world, can stop an idea whose time has come.
– Victor Hugo

Is NoSQL the future?
No, the future is Polyglot Persistence!
– Martin Fowler

Most people find the concept obvious, but the doing impossible.
– Alan J. Perlis
Polyglot Persistence

- Using multiple data storage technologies for an individual application,
- chosen based upon the best way data can be stored and retrieved for the application.
• Leverage from highly specialized solutions which
  – use highly optimized data storage formats and
  – provide great query performance
• Mostly all NoSQL solutions available as Open Source Software
Bane

• **Architecture:**
  Increased overall complexity

• **Implementation:**
  Impedance mismatch between database approaches on data structuring, queries, etc.

• **Operations:**
  New challenges like separate backup/restore approaches, different high-availability models, etc.
Fundamental Architecture Principles
Methodology

1. Use Cases & Requirements: What data management requirements do you have?
2. Technology & Products: What database solutions you want to use?
3. Architecture & Approach: How can you integrate the database solutions?
Step 1: Use Cases & Requirements

- **Basic:**
  - Entity CRUD Operations
  - Query Entity by Id
- **Advanced:**
  - Query Entity by Example (Fields)
  - Query Entity by Relationships
  - Transient Data (Sessions)
  - Mass Data (Logging/Tracing)
  - Bulk Data (BLOB)
  - Time-Series Data
- **Challenging:**
  - Full-Text Searching (FTS)
  - Hierarchical Data
  - Graph Data
## Step 2: Technology & Products

<table>
<thead>
<tr>
<th>Database Type</th>
<th>Use Cases / Requirements</th>
</tr>
</thead>
</table>
| Relational      | Entity CRUD  
Entity Query by Id  
Entity Query by Relationships |
| Document        | Entity CRUD  
Entity Query by Id  
Entity Query by Example  
Bulk Data |
| Graph           | Hierarchical Data  
Graph Data  
Entity Query by Id  
Entity Query by Relationships |
| Wide-Column     | Mass Data  
Time-Series Data |
| Key-Value       | Transient Data  
Time-Series Data |
| Full-Text       | Full-Text Searching |
## Step 2: Technology & Products

<table>
<thead>
<tr>
<th>Database Type</th>
<th>Database Product (preference)</th>
<th>Database Product (alternatives)</th>
</tr>
</thead>
<tbody>
<tr>
<td>Relational</td>
<td>PostgreSQL</td>
<td>CUBRID, MySQL</td>
</tr>
<tr>
<td>Document</td>
<td>MongoDB</td>
<td>ArangoDB, CouchDB</td>
</tr>
<tr>
<td>Graph</td>
<td>Neo4J</td>
<td>ArangoDB, OrientDB</td>
</tr>
<tr>
<td>Wide-Column</td>
<td>Cassandra</td>
<td>Hypertable, Hadoop/HBase</td>
</tr>
<tr>
<td>Key-Value</td>
<td>Redis</td>
<td>Riak, Memcached</td>
</tr>
<tr>
<td>Full-Text</td>
<td>ElasticSearch</td>
<td>Solr</td>
</tr>
</tbody>
</table>
Step 3: Architecture & Approach

• There is no "one size fits all" integration approach.
• Four reasonable approaches are possible in practice:
  – Multiple Lanes
  – Polyglot Mapper
  – Nested Database
  – Omnipotent Database
Approach 1: Multiple Lanes

Data Facade (DF) component of application dispatches into multiple separate persistence lanes.
Approach 1: Multiple Lanes

Data Facade (DF) component of application dispatches into multiple separate persistence lanes.

<table>
<thead>
<tr>
<th>pros</th>
<th>cons</th>
</tr>
</thead>
<tbody>
<tr>
<td>very simple to implement</td>
<td>data should be domain-specific pre-partitioned</td>
</tr>
<tr>
<td>works for all types of databases</td>
<td>cross-lane queries are hard to implement</td>
</tr>
</tbody>
</table>
Example: Mission Control

- **Domain:** Employee Mission Scheduling
- **Approach:** Polyglot Persistence via Multiple Lanes:
  - PostgreSQL RDBMS
  - MongoDB DMS
  - Apache Solr FTSDB
- **Benefits:**
  - BLOBs externally
  - Full-text search and auto-completion over all fields
Example: Mission Control
Approach 2: Polyglot Mapper

Data Mapper (DM) component of application handles multiple databases in parallel.
### Approach 2: Polyglot Mapper

Data Mapper (DM) component of application handles multiple databases in parallel.

<table>
<thead>
<tr>
<th>pros</th>
<th>cons</th>
</tr>
</thead>
<tbody>
<tr>
<td>single object model</td>
<td>just a few mapper products support RDBMS+NoSQL</td>
</tr>
<tr>
<td>cross-lane queries possible</td>
<td>feature-set is just intersection of database types</td>
</tr>
</tbody>
</table>
Approach 3: Nested Databases

Backend (BE) of primary database transparently maps secondary database(s) into its own data model.
Approach 3: Nested Databases

Backend (BE) of primary database transparently maps secondary database(s) into its own data model.

<table>
<thead>
<tr>
<th>pros</th>
<th>cons</th>
</tr>
</thead>
<tbody>
<tr>
<td>polyglot persistence invisible to entire application</td>
<td>still too less database products support this approach</td>
</tr>
<tr>
<td>cross-database queries possible</td>
<td>feature-set is partially constrained</td>
</tr>
</tbody>
</table>
Approach 4: Omnipotent Database

Database uses multiple Backend (BE) storage engines in parallel: relational and non-relational.
Approach 4: Omnipotent Database

Database uses **multiple Backend (BE) storage engines** in parallel: relational and non-relational.

<table>
<thead>
<tr>
<th>pros</th>
<th>cons</th>
</tr>
</thead>
<tbody>
<tr>
<td>polyglot persistence invisible to</td>
<td>only a few database products support this approach</td>
</tr>
<tr>
<td>entire application</td>
<td></td>
</tr>
<tr>
<td>holistic backup/restore is possible</td>
<td>feature-set is very constrained</td>
</tr>
</tbody>
</table>
Revolving World Order...

- **PostgreSQL** RDBMS nowadays provides XML, JSON, Key/Value (hstore) and FTS (GIN)
- **MongoDB** DMS recently got FTS support.
- There are Data Mappers which store whole object graphs into Key/Value stores.
- ...
My Recommendation

1. Think about your major challenges and make sure you really have one which is persistence related.
2. Try to use a single RDBMS or NoSQL database.
3. Leverage from Polyglot Persistence. But always keep the number of databases as small as ultimatively possible!
My Recommendation

• Approach 1 (Multiple Lanes) for Transient Data like Sessions and for „fire-and-forget“ Mass Data like Logging/Tracing Messages

• Approach 2 (Polyglot Mapper) or Approach 4 (Omnipotent Database) for Entity CRUD

• Approach 3 (Nested Database) for foreign but partially locally linked data like LDAP lookups
Thanks...

- Thank you for listening!
- Do you have any questions?
- Contact me under: ralf.engelschall@msg-systems.com