

What are the Options for Storage?

Overview

There is a wealth of knowledge on the subject of persisting object data. One useful general reference is the [Barry & Associates site](#)¹, another is [Scott Ambler's site](#)². Here we try to cover just some of the major ideas, in rough order of priority. To summarise at the outset, we need to consider issues such as:

- why not use an object database?
- do the data need to be retrievable from the database by software written in other languages?
- what granularity of data needs to be queryable?
- can we use *openEHR* paths to help?

Object databases and persistence frameworks

One option may be to forget all about relational databases for your persistence, depending on whether you have constraints in your deployment environment on what kind of database or persistence mechanism you are allowed/encouraged to use. The attraction of object databases and other native object mechanisms is that you don't have to think too much about how your data fits the database - because there is no semantic gap between your objects and the database. If an object database or framework satisfies all the needs of the service you aim to provide then this is a good option. You have to carefully consider all your requirements and assess them against the product you are considering. Issues to consider...

- Adding an object database to an existing environment means adding more database administration, including start, stop, backup, archive and other operations, most likely in a tool that existing sysadmin / operations staff have never seen. Make sure the overhead is acceptable. An object persistence framework probably won't be visible at all to such people.
- Some object databases and most object persistence frameworks store data in native object form, e.g.. Java objects are stored in native binary form, only retrievable by the same software and instantiable as Java objects. This may be fine, but you need to be sure.
- What is the finest grain of query you need to be able to do? There is probably no point in storing data smaller than this granularity.

An **object persistence framework** is typically a fairly lightweight library that provides: a persistence API, a method of persisting data to disk, and a smart cache. The API is typically of the form where calls like `store(an_object)` can be made, where `an_object` is the root object in a network of objects that together make up a whole top-level structure. Object persistence frameworks don't usually provide all the session management, querying, security, and transactional power of full database systems. They may or may not be scalable to large numbers of users, in may be more oriented to client-side persistence rather than server-side persistence. Examples for Java include: [db4o](#)³.

An **object database** on the other hand is a proper scalable and secure database management system that supports querying as well as persistence - in other words, like a relational databases system, except that it deals directly with objects rather than tables. Usually some object-flavoured SQL will be supported. Example products include [Matisse](#)⁴ (a language-neutral database with SQL querying). There are also

1. <http://www.service-architecture.com/object-relational-mapping/>
2. <http://www.ambysoft.com>
3. <http://www.db4o.com>

clinical information systems based on object databases such as [InterSystems Cache](#)⁵ and [Jade](#)⁶. [Zope](#)⁷ is a Python-based object database that is quite widely used behind active websites and has been used in health information systems, e.g. FreePM, [OIO](#)⁸.

Object/Relational products

An object/relational (O/R) product is one that ultimately relies on an underlying relational database to store the data but does all the hard work of turning objects into relational form to write into the database. From the programmer's point of view, it may look just like an object database. The advantage of this approach is that it allows you to use an existing relational database in your environment that is already required for some other purpose. O/R products solve the problem of performing the object/relational mapping in a generic way, but they don't *a priori* know anything about your data. In particular, they don't know about what the patterns of querying are, where the business object boundaries are, or anything else. Some products may allow such things to be specified.

The default situation will be that using an O/R product on a typical object model over a relational database will result in numerous tables and extremely fine-grained object storage and retrieval, with the consequent performance penalty. Most likely, an O/R product will not know about business object boundaries and will do the same thing as an object database with a naively designed object model: store and retrieve everything reachable by reference-following. Avoiding these problems means at a minimum reducing the granularity of the objects being stored; see below.

Examples of object/relational products include: [Apache ObjectRelationalBridge \(OBJ\) for Java](#)⁹ and [DataObjects.NET](#)¹⁰.

Relational databases

Object data can be directly stored in a relational database, but the schema design is a greater issue. If the intention is that schema is a derivative of the object model - i.e. the "classical" approach to mapping ([typical strategies](#)¹¹) then the schema design may not be trivial. This kind of schema design is what many of the O/R tools try to automate and/or hide. However, other strategies are available, including one very interesting one which is possible due to the paths in *openEHR* data.

See this [wiki page](#)¹² for an approach called 'node+path' which shows how a relational database could be used to store path-based archetyped data such as that found in *openEHR*.

Fields

-
4. <http://matisse.com/>
 5. <http://www.intersystems.com/cache/>
 6. <http://www.jadeworld.com/>
 7. <http://www.zope.org/>
 8. <http://www.txoutcome.org/>
 9. <http://db.apache.org/ojb/>
 10. <http://www.x-tensive.com/Products/DataObjects.NET/>
 11. <http://www.agiledata.org/essays/mappingObjects.html>
 12. <http://www.openehr.org/wiki/pages/viewpage.action?pageId=786487>

Name	Value
FAQCategory	Persistence Design