

When a development project requires a database, it's all too easy to reach for the 'usual' relational database servers; after all, you're going to be storing names and addresses, order processing data, and statistical information for analysis. Then comes the moment when you're going to write code in your chosen programming language to work with that data. You're almost certainly going to be using an object-oriented language, so you're then into objects, classes, inheritance, and your relational database structure doesn't really fit that well; you have to map tables to objects; try to force SQL queries to act like methods. It's not easy, and you lose many of the advantages of object-oriented programming – inheritance, encapsulation. Add in the increasingly important requirement to be able to work directly with persisted XML format data, and the relational model starts to creak at the seams.

Caché, from InterSystems, takes a different approach. It's marketed as a 'post-relational' database. It's a multidimensional database that works with objects as well as it handles 'normal' relational data. All data stored in Caché is stored in sparse multidimensional arrays. The One of the main intentions behind this is to eliminate the processing overhead caused by the need to manage the joins between tables in relational databases. The real attraction of Caché, however, is the ability it gives you to simultaneously work with the same data as both 'ordinary' relational data (it's a SQL 92-compliant relational database) and data stored as objects. Objects created in Caché support encapsulation, multiple inheritance, polymorphism, embedded objects, references, collections, relationships, and BLOBs. In addition to this strong underlying database engine, Caché comes with a good development environment called the Caché Studio that eases the path of application development through the use of wizards to guide you through creating objects, databases and entire applications.

One of the major advantages of

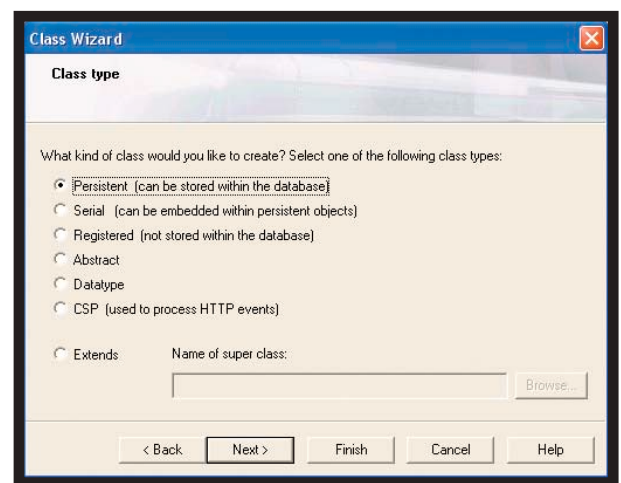
# Caché gains strength

## Moving beyond the relational model

BY KAY EWBANK

developing with Caché is the ease with which you can work with data, no matter what its underlying type. Caché calls the technology that makes this possible the Unified Data Architecture. This lets you work on data as either objects or tables, so you don't need to have two sets of data definitions, or to specify how data in tables maps to data in objects or vice versa. This makes development much faster and simpler.

One potential drawback of using objects to store your data is that your users probably have favourite report writers and query designers that they are happy with. Fortunately, when you define a database object class in Caché, behind the scenes you get SQL access methods to work with that data, so any applications and utilities that use SQL as their access method can work directly with the data as though it were sitting in a normal SQL table. Going the other way, if you've got an existing relational database, and you import its DDL (Data Definition Language) definition, Caché generates an object description of the



Choosing the class type for a new class in Caché Studio

data, so you can start working on it using the object methods in Caché.

Caché is also an excellent way to create database applications for the Web. You can literally take a Caché 'table' (to use the traditional name), work your way through a simple wizard, and end up with a Caché Server Page. Move to a suitable Web development environment, and you can be turning that server page into a Web form and viewing your data in minutes. It is very good when used with MacroMedia DreamWeaver, where the Caché classes appear as DreamWeaver Extensions, but you can work in pretty much any Web development environment.

### Security

Keeping data safe is obviously important, and Caché offers multi-level security. You can assign privileges to data, programs, services and utilities within Caché, and set the privileges for users, roles, and applications. Security is managed through the Caché Management Portal (a Web-based portal where you can control most aspects of

“ Caché's solution is to use transactional bit-map indexes. These are multidimensional data structures, so you don't have as much 'empty' space in the index, and therefore don't use as much disk space.”

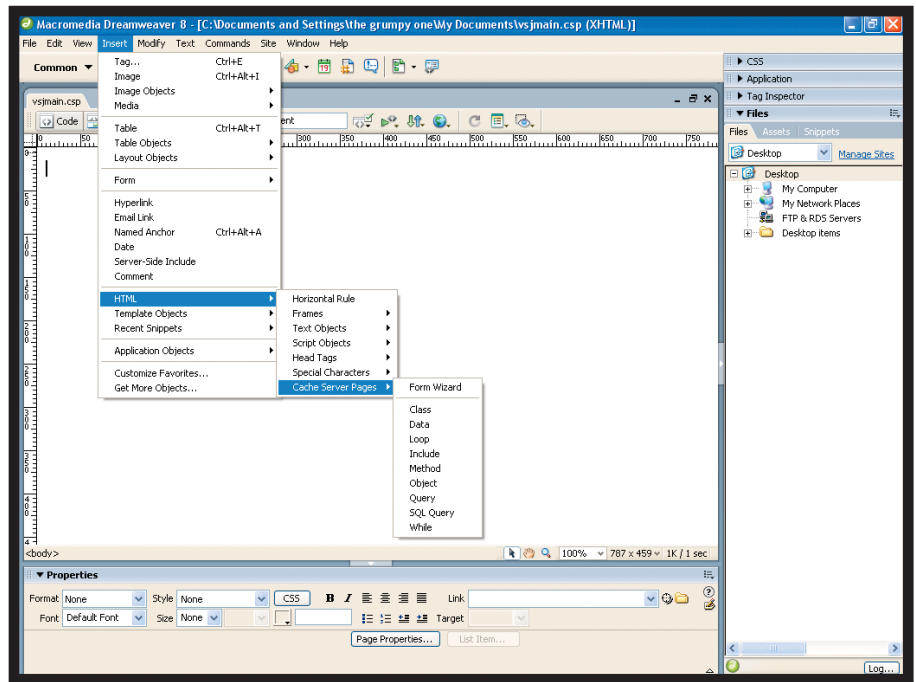
Caché). You can also choose the level of authentication, ranging from no authentication to the use of Kerberos authentication. You can encrypt your databases, or parts of databases down to block level, and both data and indexes are encrypted.

## All this and speed too

In a database developer's mind, any move away from straightforward relational data is equivalent to saying 'runs like treacle'. This is not the case with Caché. It actually performs very well – faster on relational data in many cases than 'traditional' databases. The reason is, according to InterSystems, that there's no need to join multiple tables to carry out transactions, so the underlying record retrieval is much faster. It also supports a technology called transactional bit-map indexing.

To understand this, you need to start from the way a traditional set of indexes work. In a relational database, you create an index for the columns or combinations of columns that you think are going to be useful for finding data – the product ID code and the product name, perhaps, or the customer ID, surname, city. The more indexes you have, the bigger the overhead when adding or editing data; if you need to look for data in a column without an index, you'll wait a long time for the results. Under the covers, each index is a collection of lists. For each value in a particular column, there's a list of the row IDs in the database table that have that value.

A bit-map index is more like a grid.



If you want to create a standard Web application, Caché can be used with DreamWeaver, which recognises Caché Server Page objects

There's separate bit-map for each possible value of a column, with one bit for each row that is stored. A 1 bit means that the row has that value for the column. This means that complex queries can be defined by carrying out Boolean operations such as AND or OR on the indexes without the need to search the whole database. The response time for such indexes is well known to be excellent. The drawback is that they are huge, taking up lots of disk space; and they are very time-consuming when you're adding or updating data.

Caché's solution is to use transactional

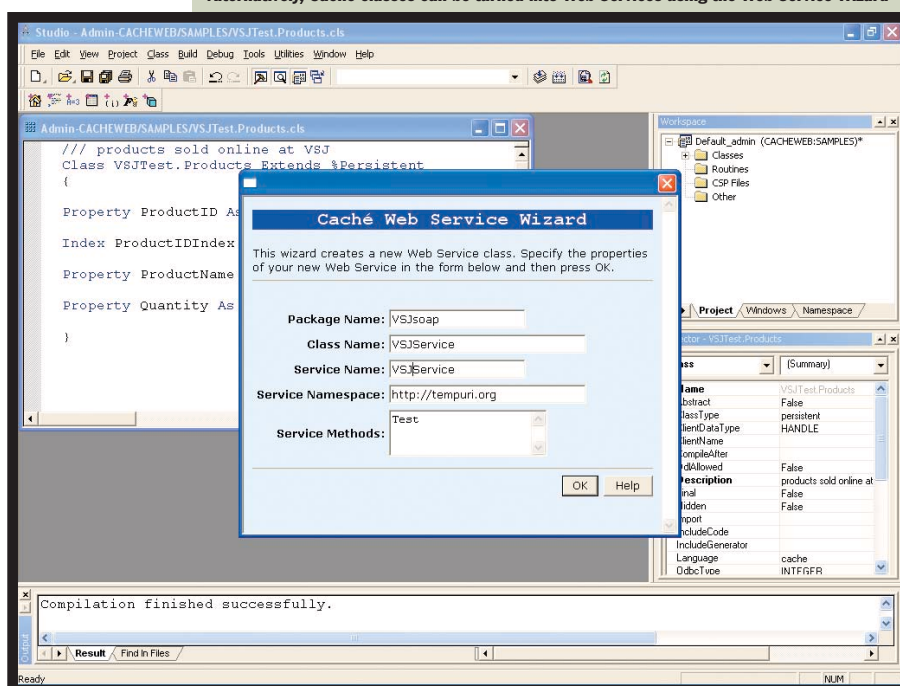
bit-map indexes. These are multidimensional data structures, so you don't have as much 'empty' space in the index, and therefore don't use as much disk space. Updating is also faster, so that the indexes can be used even in transaction processing applications.

Another way Caché achieves its impressive performance is by the use of an Enterprise Cache Protocol (ECP). This manages a multi-level distributed cache on and between Caché servers. When a client requests data, the application server first attempts to provide the information from its local cache. Whenever this is possible the load on the actual data server, and the traffic between the data server and the application server, is reduced. If data has to be retrieved from the data server, the whole database block is requested, so that subsequent data requests will probably be met from the newly retrieved data.

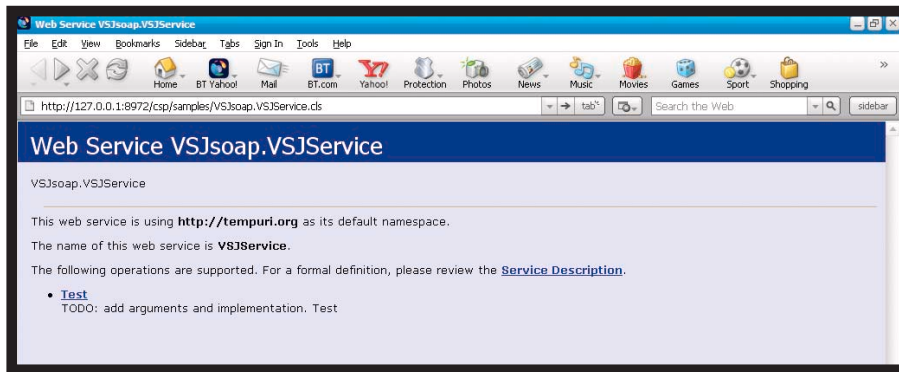
## Developing in Caché

The main environment for developing in Caché is Caché Studio. This is an easy to use and attractive IDE that will provide no shocks or surprises to developers used to Visual Studio or similar environments. When it comes to actually writing code, the recommendation is that while you could use external languages – Java, C++ or whatever – you will get the best performance and scalability if you write your business logic scripts in Caché itself. There are two languages on offer – Caché ObjectScript and Caché Basic. ObjectScript is the original Caché scripting language, and it is

Alternatively, Caché classes can be turned into Web Services using the Web Service Wizard



“The way in which we interact digitally, and the changes that fact engenders, mean we need more than SQL.”



And after a quick compile, the Web Service can be viewed and used

You can choose where the functions you use in your CSPs (Caché Server Pages) are executed – on the Caché data server or in the Web browser. You can also specify that certain events in your pages can trigger procedures to be executed, either on the client or the server, then update the current page without completely repainting it by returning code to be executed on the browser to make the Web application faster and more responsive.

### Other matters

Other things you ought to know about Caché: you can download a single-user evaluation version free from the InterSystems Web site, so you can try it out and see whether it solves your problems and lives up to its reputation before buying a copy. It's also worth saying that the tutorials that are included with the software are excellent; they're clear and easy to follow, show you all the stages, and take you through the creation of all the main types of database you're likely to want to work with, as well as showing you how to set up Web services and Web applications. There are good sample databases too.

### Conclusion

We've looked at Caché before in *VSJ*, and it has always seemed a good product. In the past, the 'but' was whether a generation of database users brought up on the assumption that you had to use a relational database could see beyond this view to the advantages offered by Caché. Increasingly, the way in which we interact digitally, and the changes that fact engenders in user requirements, mean we need more than SQL. Caché is easy to use, fast, robust, and handles object data and relational data equally well. Give it a try – you'll be impressed.

■ Kay Ewbank, Editor of *Server Management* magazine, is a highly experienced database analyst who has followed the development of database technology from dbase through to today's SQL servers.

■ InterSystems UK  
0800 587 6544  
[www.InterSystems.co.uk](http://www.InterSystems.co.uk)

compatible with InterSystems' other products such as Ensemble. It lets you mix your data access methods so you can look at your data as objects, as relational tables (using SQL), or as multidimensional arrays. If you're familiar with Visual Basic, you might prefer to use Caché Basic. This is similar to VBScript, and was added to provide a familiar environment for VB programmers. As a postscript, the upcoming version of Caché extends its Basic implementation to run MultiValue (formerly known as PICK) applications more or less natively.

### Developing outside Caché

While the Caché Studio is a really nice development environment, not everyone will want to move outside their 'normal' programming language, and it's easy for other applications to be used to work with data held in Caché. You can use either ODBC or JDBC as the data driver, and the underlying data is then accessible to your reporting tool, spreadsheet, or whatever. This access is managed using a further element of Caché, the Caché Gateway.

If you want or need to write code to work with Caché and don't want to use either of the internal languages, you can make use of the Caché classes from other languages. Caché classes can be exposed as Java, COM, or C++, and you can also choose to expose the classes as EJBs. There's a managed provider for .NET that gives yet another route through to Caché, and gives relational access to data using the ADO.NET API, as well as native object access to data using auto-generated proxy classes. Caché has particularly strong support in this area for Java developers, as your Java persistence classes are automatically generated for you, so you have Plain Old Java Objects (POJOs) with relational access with no manual mapping.

### Web Services

An increasingly important way that Caché is

being used is via Web Services.

There are two three reasons for this. Firstly, it's very easy to set up a Web Service from any Caché class – there's a Web Service Wizard that you click your way through, and without any extra effort, there's your working Web Service. The second reason for the increase in use for Web Services is simply because Caché is often encountered as an "embedded" database, and the creators of such databases tend to be responsive to what their customers need. Finally, many web applications implement an object model to which Caché provides a native, efficient answer. Web Services make it easy to provide the facilities customers are looking for in the environments they want to work in, and it's easy to make existing Caché applications Web Service enabled, so Caché developers are filling that need.

Because support for Web Services is built into Caché, you don't need any additional middleware. All you have to do is to have a Caché class in which you've specified some of the methods as "Web Methods". These are then automatically made available as Web Services, and Caché also automatically generates the WSDL document for your Web Service, and takes care of making the document available to prospective client applications.

The great news for 'traditional' developers is that you can make your applications Web Service friendly with very little work, or indeed with very little knowledge about Web Services. Once you've set up the Web Service everything happens automatically. When a SOAP client makes a request, it is handled by the Caché SOAP Server, which unpacks the SOAP message, converts all parameters to their proper Caché representation, and invokes the appropriate Web Method. The Web Method is executed – maybe querying the database, for example. The response is created and passed back to the SOAP Server. This packages the response, and sends it on its way back to the client.