Can we store Java objects without using SQL? Yes, we can!

Jose.Ruperez@InterSystems.com
Java Certified Programmer, Caché Certified Expert

We programmers have a problem. Every time a program is written that requires data storage, we must use a database. For the past few decades, this has been a relational database, which means talking to the DBAs to design a database model that reflects our object model. Therefore, two models must be maintained.

To solve the problem, programmers have spent a lot of time implementing the database layer using SQL. Functions have to be written to store objects using the INSERT and UPDATE statements. More functions have to be written to retrieve data from the database using SELECT statements, mostly using complex JOIN statements to combine data from many tables so that our objects can be instantiated in memory. This whole process takes too long and does not add any value. If anything, it adds complexity to our code and slows it down. We refer to this problem as the "object-to-sql mapping problem."

Since the inception of Java in the 1990's, many have tried to come up with a viable solution to this problem. The goal has always been to reduce and automate the time it takes to implement the database layer while at the same time improve the speed of the program. However, should we continue this line of thinking or is there another approach?

There is a solution to the "object-to-sql mapping" problem, though not very well known: InterSystems Caché. Caché is a high performance object database, developed by InterSystems. The Java plus Caché combination is the perfect formula for delivering breakthrough applications.

How does it work? Caché uses reflection to look into the Java objects and generates the database schema automatically. Caché provides an API for you to store and retrieve your objects. All you need is to rewrite your database layer using this API. The objects are ultimately stored in

multidimensional sparse arrays, which is the single most important reason for the high performance results.

```
System.out.println("Storing...");
Event event = persister.getEvent(className);
// Get Data ...
System.out.print("System: ");
String s = in.readLine();
System.out.print("Username: ");
String u = in.readLine();
System.out.print("Password: ");
String p = in.readLine();
PassRecord record = new PassRecord(s,u,p);
event.store( record );
event.close();
```

1. Sample Store Code

```
System.out.println("Searching...");
// Get System Name
System.out.print("System: ");
String s = in.readLine();
// Search via SQL
String sql = "SELECT * FROM PassRecord WHERE system = '"+s+"'";
EventQuery<PassRecord> query = null;
try {
    query = persister.getEvent(className).createQuery(sql);
    //query.setParameter(1,s);
} catch (java.lang.Exception e) {
    System.out.println("createQuery() failed:\n" + e);
}
query.execute();

// Iterate through the returned data set, printing and deleting each event
EventQueryIterator<PassRecord> iterator = query.getIterator();
PassRecord currentEvent = new PassRecord();
while (iterator.hasNext()) {
    currentEvent = iterator.next();
    System.out.println("Retrieved Username=" + currentEvent.user + " + Password=" + currentEvent.pwd );
}
query.close();
```

2. Sample Retrieve Code

As a bonus, even though Caché works with the object model, an sql view of your data is also available. It is easy to download a free copy of the software and test your application with InterSystems Caché. You will be astonished by the results and most likely you will say to yourself, “Why didn’t I hear about this earlier?”

Apart from the object-to-sql mapping problem, InterSystems Caché can solve many other common programming concerns such as performance and scalability. With Caché, it is possible to sustain insert rates of over 100,000 records per second and scale to hundreds of nodes.

InterSystems Caché is a different sort of database, very robust and with over 30 years of experience in many industries throughout the world. For more information, visit: www.InterSystems.com/download