

B2B Analysts

Workday Rising

"From an in-memory database...that works completely differently from a relational system, we can expect wonders."

-- Hasso Plattner

The torrent of good will and affection that has greeted Dave Duffield's new entry into the applications market has had one odd effect: it has obscured the importance of this company and its product.

Not that the good will is misplaced. It is, after all, a great story.

Dave Duffield and Aneel Bhusri, still smarting from the wounds inflicted by Oracle, decide to start a new applications company. It will be PeopleSoft risen from the ashes: same culture, same insouciant attitude, even the same starting point (HR). They'll make it a true hosted product; they'll use all their previous knowledge and build it right. It's hard not to root for them.

After all, it would give us all a thrill if someday, years from now, Dave appeared once more in the lists and called Larry out.

You know, if we could just get Harrison Ford to play Dave Duffield, the rest of the casting would surely fall into place. We could start shooting Summer '09 for a Christmas release.

When I first heard about Workday, the more it sounded like a movie plot, though, the less I liked it. Whatever Dave's abilities, I felt, you don't succeed in the software business by re-fighting the old battles.

Everybody wants to get a chance to correct all their past mistakes--and in the movies they get a chance to. But in the software business, especially, the trick is to do it right the first time, because you don't get a second chance.

In any case, I didn't see it as a matter of personalities; the issue for me was the underlying technology.



David N. Dobrin
President
B2B Analysts,
Inc.

David is a recognized expert on ERP, supply chain, and collaboration applications. Before founding B2B Analysts, he was Chief Business Architect at Benchmarking Partners. While at Benchmarking Partners he helped companies develop coherent and workable application strategies, and also sat on the original CFAR committee, where he helped develop what became the CPMR standard.

Reprinted with permission from the original analyst note, "Short Forms #30; Workday Rising" which was published on June 15, 2007.

B2B Analysts research is independent. This perspective was not funded by the vendor mentioned in the report.

If you remember, I argued last October, in "The Steam Loom Problem," that it's usually not economically practical to replace (not upgrade, but replace) infrastructure software with something on a new platform that still does basically the same thing, no matter how many cupholders have been added. If that argument is right--and naturally, I think it is--even A1S and Fusion AS can't replace R/3, 11i, One World/World, or PeopleSoft. If they can't, then how could Workday ever do it?

To justify replacing the now-aging infrastructure software that most of the world's companies have installed, I believe you need software that offers some brand new and significant innovation. What Workday seemed to be doing, which was hosting a somewhat improved version of the old PeopleSoft product, just didn't seem to be enough. I wished them well, and I welcomed the return of a software company that people actually wanted to work for. But I didn't see anything like the last scene of Rocky happening any time soon.

Well, I've now had a chance to look at Workday the company and Workday the product. I've made the pilgrimage to Walnut Creek, gotten a technical briefing, and attended one of the early roadshows.

Guess what. That innovation I want, the one that might actually give people a reason to rip out their zillion-dollar HR or financials installation(s) and replace it with something that works significantly better. The one that nobody's been able to come up with for 15 years? They might just have it.

If they do, Workday will be the most interesting and important company in the applications space since Salesforce. And, over the long run, it has the potential to transform that space.

After making a statement like that, which regular readers know is completely out of character, I certainly don't want to try your patience too much.

So in this piece, all I want to do is lay out what the innovation is and why it's important. In the next, I'll talk about what Workday has and has not done so far with this innovation, what the initial reaction among customers is, and what Workday has told me they're planning to do next.

The Innovation

As you'll see, though, even describing the innovation isn't all that easy, either for me or (I've found) for Workday.

With innovations, you can start in one of two places, with the innovation itself or with the problem it's trying to solve. Let me start with the first, move to the second, and then go back and show why the first solves the problem.

The innovation itself is a hosted, non-relational (or object-oriented) database, on which Duffield, Bhusri and company have chosen to write a suite of business applications.

The components of this innovation have been around for a while.

In-memory databases have been around for 8-10 years, I think, and so have object-oriented databases. Hosting for business applications still strikes as innovative, but conceptually, it's been around for a very long time, and even the Salesforce innovation on that innovation, the multi-tenant hosting model, is nearing its second decade. Business applications? Well, they're about as old as snuff and anti-macassar.

The combination, however, is relatively new; in fact, it was barely even conceivable five years ago. And what Workday is doing with this combination is entirely new.

The Problem

To understand the problem that they're addressing, you have to understand something about how all the modern business (ERP) applications work. All are built on top of a relational database that persists (stores) the data on a disk. All of them, I think it's fair to say, were designed to run on this kind of data store and could not have existed in their current form had the relational database not been invented. Indeed, much of their success in the marketplace was due to the fact that the relational model gave them a clear edge over applications that used other data storage technologies.

A relational database consists of a number of highly structured tables which hold a number of structured fields (think columns in the table). Much of the work of building an application on a relational model lies in defining those tables and the relationships between those tables. In an HR system, for instance, the heart of the system will typically be an Employee table, which will have fields like First name, Last name, Middle name; multiple address fields; a job category field, etc., etc. In the same application, depending on how the application designer thinks, there might also be another table, called "Job Category," whose fields contain the name of the job category, pay range, job description, etc., etc., etc.

The application itself basically stores information in these tables and retrieves this information. The way it does this necessarily uses the relational structure. So, for instance, to find out about the employee's pay range, you look up his or her record in the Employee table, find a value in the Job Category field, and then use that value to find the corresponding record in the Job Category table. Once you find that record, you look in the Pay Range field(s), et voila!

This may sound convoluted, but the relational model was a huge advance over what went before. The structure of the tables guaranteed a high level of integrity; there were also big advantages in speed and in cost of storage.

Unfortunately, though, when it came to writing business applications, this huge advance had two obvious weaknesses. One, if you wanted to get information that required that you traverse a lot of tables, you could do it, but it would be complex, and it would be dog slow, so slow and complex that you probably wouldn't bother. Two, whatever you decided to use for tables and whatever you decided to use for data in the tables, you were going to be forced into fairly difficult and complex tradeoffs, which you were going to have to live with.

This last weakness was poorly understood twenty years ago, and I'm not sure it's much better understood today, but it is exceedingly important. It is what is responsible for all the complaints about "inflexibility" and most of the complaints about "quality of fit" that have been levelled at every single vendor of ERP applications (not just SAP) over the past twenty years.

Let me try to explain it with a simple example. This may not convey how gut-wrenching and deep the problem is, but it will give you an idea of why it occurs.

Let's say that in the HR application above, the designer wants to serve large companies with several divisions. In each division, some aspects of a job category, such as pay range and job description, are unique. To serve this need, the designer decides to put a new Division table into

the database and a Division field in the employee record. Each employee is now assigned to a division. He also creates a new table that holds the unique components of the job description that each organization wants to create.

Now, if you want to look up information about the employee, including both the common parts of the job category and the unique pay range/description, you look up the employee, find the organization code and job category code that belong to him or her, look up the common job category stuff using the job category code, and look up the unique parts in another table, using both the job category code and the organization code.

(Database purists might not like this way of doing it, but if you want to understand the weaknesses, this is a useful way of describing it.)

In this example, the first weakness should already be pretty clear.

Already, you're doing a whole lot of table-traversing. And all this traversal is disk-bound. You have to go down to the disk, find the table, pull up the data until you find what you want, go to the next table, etc. So of course it's slow.

You can only see the second (and more important weakness) if you start thinking about how this way of describing what the employee does differs from one's "natural" way of thinking about it. Notice, for instance, that the employee can only belong to one division and thus can have only one pay range. But in real life, employees sometimes belong to more than one division; sometimes, the job is even shared. It's a natural thing to do, and it happens. But in the design that we've made up, job sharing simply can't be represented.

Or consider what would happen if you set up a system like this and decided to change your divisional structure. You would have to rebuild the division table and the unique job description tables, and you'd have to change the division values in every employee table.

You can do that, of course--software can do anything--but if you do, you've created another ugly problem: you've lost the way the system used to look.

Your reports will be accurate about what is in the database today. But you won't be able to go back and look at how it used to look before the change.

Ugly, even for four tables, right? Well, think how complex problems like this (which inevitably come up in a relational database) can be when you have "tens of thousands of tables," as you do in SAP.

As a practical matter, people just deal with problems like these. Yes, there's a lot of complexity and a lot of inflexibility that are artifacts of the way the applications designed for relational databases are built and implemented. And yes, you live with it. In the example above, you just do your employee-sharing outside the system; you spend the money on rebuilding the database if that's what you have to do, and if somebody asks you for a report on how the company used to look, you find some way of losing the e-mail. That's how it is, and that's what you do.

The In-Memory Database

Or do you?

I'm going to make a big leap, here, so bear with me. Imagine what would happen if you were designing these slow, complex, and inflexible systems and were able to push a whole lot of the operations that made your database so slow and inflexible up off the disk and into computer memory.

Essentially, you'd be able to relax some of the constraints that traditional relational databases exert. Just which constraints might well be up to you; it would depend on what you wanted to accomplish.

This prospect is getting a lot of people in the industry excited.

If any of you listened to Hasso Plattner's keynote at Sapphire (from which I take the epigraph above) or heard him at Software 2007, you can see that he's grasped the idea and wants to take it places. At SAP, as I understand it, in-memory, non-relational techniques are part of what is responsible for the blazing speed of the BI Accelerator.

One reader of this newsletter argued to me (I think correctly) that one of the best-known and most advanced uses of in-memory databases is at Salesforce.com. To get around the limitations of the relational database that they use underneath their system, he says, they had to write a whole bunch of code (executed in memory by definition) that would speed up database access and reduce the cost of traversing multiple tables.

Oracle, too, is clearly interested in in-memory databases; they recently bought Times Ten (an in-memory database company) and the blogosphere indicates that they are looking at uses of in-memory technology that go well beyond what Times Ten provides.

In all three of these cases, the constraint that people seem to be focusing on is the dog-slowness of traversing the tables. This is a stretch, but essentially, they seem to be saying, "If we put all those tables into memory, then moving from one to another and assembling the information will be a snap."

In each of these cases, however, the complexity and inflexibility that are artifacts of the relational model aren't being addressed. Even in memory, if you still have one primary organization code, that will make belonging to two organizations impossible.

What seems to make Workday unique is that they've decided to use the in-memory technology to address those issues.

To see what I mean, let's return to the example. If you were designing the system without any tables at all (or at least without relational tables), you'd probably want to do something like this. Instead of thinking of an employee as "belonging" to one organization, you'd want to let the users of the system set up multiple organizational structures and multiple relationships to the organization structures and allow each employee to "belong to" all of those structures. So if an employee is paid by one organization (a financial organization) and is seated in a different organization (a geographic organization) and does work defined by a still-different organization (a reporting organization)--which is usually what you mean when you say an employee is shared--the

employee record can be viewed through each of those facets, and depending on which facet you're looking through, the correct information relative to that facet will be shown to you.

This is what anybody would really want from an HR system, of course; it is the natural way of looking at the record (that is, the employee).

And this is, I believe, what Workday is trying to provide.

How do they do this? Well, this is where the hand-waving really starts.

Basically, I don't have a clue, and I think that's partly because I wasn't really told, but I'm also sure that if I had been told, I wouldn't have had the wherewithal to understand it.

Here is what I do know, and even this is pretty startling to those of us brought up in the relational world.

Workday runs on an in-memory database (64 bits, at least 1 TB of memory); it uses a relational database (MySQL) only for persistence (that is, to store the information in case the in-memory database goes down). This persistent database has exactly three tables, one for data, one for metadata, and one for instructions. Yes, that's right. All data--the employees, the positions, the organizational structure, and even the layout of a screen--is stored in the same table.

At Workday, then, all the work of identifying what a record means is done in in-memory. When a data record is hauled out, something up there in that TB of memory figures out that this is the record that would belong to the employees table in a relational database, that the first blob of data is a last name, the second a first name, etc., etc.

Obviously, what you can do with this record depends on how you wrote those instructions. In my 2-hour briefing (Aneel was generous with his time), I was by no means able to figure everything out. But what was abundantly clear is that the Workday people were trying to explode (and I mean explode) limitations that in their previous PeopleSoft life, they and their customers had found to be particularly burdensome.

Here are some examples.

Limitation: privacy requirements. In HR applications that use relational databases, it is very difficult to give people access to the database that they need without giving them access to private, personal information stored on the database.

Workday simply gets rid of this limitation by encrypting every piece of data. (This is effectively impossible in a relational database.) Now, your access to the data is defined by your place in the organization and the privacy rules, not by what role you have in database administration.

Limitation: storing changes. As I noted above, most applications just keep the present value of most fields; they don't keep the old value and the date of the change. (SAP is, to some extent, an exception here.) They do this to make the thing go faster and to reduce complexity. But the effect is to cloud the past, shall we say.

Here again, Workday simply doesn't have the same constraint.

All changes are kept and all old values are kept, and anything can be rolled back. Effectively, for any date, Workday can provide you with the as-of state of the entire database.

Limitation: many different organizational models co-exist in most normal businesses.

Workday seems to allow many different organizational models to be embedded in the system, allows any or all of them to be changed relatively easily (but rolled back, if you want), and allows them to coexist.

Because the system is not constrained by a relational model, there doesn't have to be a single, primary organizational model, as there is in the typical ERP system. You seem to be able to effectively add any number of facets to a record and then view a collection of records using the facets that you're interested in at the moment. How do they do this? Well, it has to be in the instructions, but how they make them work I do not know.

Limitation: application developers did not grasp the limitations that their relational system imposed on users.

In Workday, though, the application is the development environment. This means that developers have to use the same strategies and techniques to build the application that the application itself uses. This sounds almost trivial, compared with some of the other things I'm talking about. But in my experience, it's just as important as any of the others. Anything that gives developers a gut-level feel for how to overcome modeling problems is going to make their work more accurate and more responsive.

Gauging the Impact

When Salesforce first came out with a system that offered clear, though limited advantages to customers, all of the competitors responded roughly as follows. "This company is completely insignificant. The product they've got is not robust, and the company is too small to build a robust product, and the advantage they offer (the delivery model) isn't an advantage, and besides, we do the same thing with our hosted applications."

Workday is just a flyspeck in the world of enterprise applications today; it has exactly 12 customers. So I doubt that anybody at either Oracle or SAP has even gotten to the point where they're trying to formulate an equally dismissive reply. Clearly, though, the same strategy could be used. You deny the advantages and claim that you're using the same technology in the same way. You brag about the improved organizational flexibility of your Fusion or A1S application, cite your 15 years of experience in successfully dealing with compliance and privacy issues, etc., etc.

Will customers be able to see through this? On the one hand, it's not a good sign that it takes me so long to explain what's so great about what they're doing; it suggests that this is an innovation that the customer base will struggle to grasp. On the other hand? Well, all I can say is that honestly, I felt for a moment when I was looking at this the same way I felt when I first saw the Xerox 'Star' computer at Xerox PARC. (The 'Star' was the precursor to the Lisa and Mac.) If this kind of advance can provoke that kind of reaction in cynical people like me, Workday will be able to fight their way through the fog and gradually convince at least the forward-looking people of how powerful their innovation is.

Here, of course, is where it is helpful to have Dave Duffield, lots of funding, a real track record, and Hollywood screenwriters (not to mention Harrison Ford) waiting in the wings. It's one thing to have some wild-eyed techie say, "This makes a difference." It's another when the Workday crew says it.

My guess is that the case can be made and it will be. If so, Workday will become a force to be reckoned with. The companies most likely to be impressed with and want the kind of application that Workday is building are large companies and growing companies, the companies that SAP and Oracle today regard as their natural, er, prey. People have said to me, often, "Well, Workday, with its hosted product, will never be able to compete with SAP and Oracle." Now, having gone behind the lines and watched them prepare, all I can say is, "Competing with SAP and Oracle is exactly what they're trying to do."

In the next piece, I want to talk about what Workday is doing so far with their innovation and how at least a few customers are reacting to it.

About B2B Analysts

B2B Analysts was founded in 2001 because we saw a need in the marketplace for analysis of business applications that focused on their value and the way they were actually used. Our mission is to provide objective, in-depth analysis that enables senior people to make accurate decisions about how to get value from business applications.