OOPSLA 2006

Panel „Objects and Databases, State of the Union 2006“

October 24, 2006, Portland, Oregon

- William Cook (Moderator), University of Texas at Austin
- Robert Greene, Versant
- Derek Henninger, Progress Software
- Patrick Linskey, BEA
- Erik Meijer, Microsoft
- Craig Russell, Sun Microsystems
- Bob Walker, GemStone Systems
- Christof Wittig, db4objects

Transcript

Brought to you by ODBMS.ORG (www.odbms.org) the Internet's most up-to-date resource portal on object database technology. An audio recording of this panel is available as a podcast on www.odbmsjournal.org.

William Cook, Moderator: Welcome, I'm William Cook. I'm moderating the panel this morning.

This is the panel on "Objects and Databases - the State of the Union in 2006". It's very important that you keep in mind that the focus of the panel is on objects *and* databases, how they interact in general. It's not specific to object databases or any particular biased approach. We want to consider the problem in general.

[Skip operational remarks]

I got interested in this because I was out in an enterprise software company trying to build applications, so I had first-hand experience of the problems of building data-intensive applications. So I am very interested in this topic. The way I look at it is that there are two aspects: How well is the persistence idea integrated with the programming language where you actually build your general purpose solutions, and then on the other one how efficient, how scalable it is. There is a tension, the level of integration and performance seem to be at odds, at least, we haven't gotten the sweet spot yet.

The other thing I wanted to say, in terms of historical things, just to give us a little context, for the object database viewpoint it is a little bit like there has been something like the A.I. winter, I don't know whether people are familiar with that, where there was a huge explosion of hype around A.I. and then there was this gigantic backslash when they didn't deliver on all the promises, and so A.I. is actually coming back now a little bit more; it is recovering from that. My hypothesis is that maybe object databases experienced maybe an object database winter, and maybe it's time the thaw is coming out there.

And for the other one that I am interested in is the whole object-relational mapping which is the other great thing that we do – trying to match objects with relational databases. We have been doing that for a long time and I am not satisfied. That's not maybe as good as it could be, maybe it could be better. I wanted to challenge the panelists to tell me that as well.

So those are just my viewpoints, I hope you all start having questions and write them down. And without any further due I like the panelists to begin.

Bob Walker, Gemstone Systems: My name is Bob Walker, I'm from Gemstone Systems. I've been with Gemstone for about 10 years. I've been dealing with object-oriented systems for closer to 15 or so years, mostly business type systems. I was gonna start by making a joke but my dog ate my notes so I'm just gonna have to adhoc a little bit here.

A lot of programmers seem to think that object orientated programming started with Java. [Laughter in the audience] And that the OR impedance mismatch is something new. Guess what? It's not! I'd like to take a quick walk down memory lane, if you don't mind. In late 80s and in the early 90s corporate systems were becoming more and more complicated. Software was starting to get very brittle, data was becoming very flawed, very inaccurate. A lot of redundancy in both the software and the data, I think we all pretty much know this litany.

Enlightened information technologists had realized there is a serious problem. Waiting in the wings was object orientated analysis design and programming. What happened? Well, object orientated programming came to the rescue by way of Visual Age Smalltalk, in many cases. There are other Smalltalk dialects that had been in use of the time. I recall a 1990s something IBM putting the stamp of approval on Smalltalk for the corporate business world, and it was turning into quite a hot market. So Java isn't the first pass at OO that corporate Fortune 1000 companies have taken. Visual Age thus gave the risk-averse corporate IT staff permission to start doing OO. People were talking about it as the next Cobol.

Typically when the proof of concept is done it's taken on as a simple application - this has been true of Smalltalk, this is true of Java. The reason I talk about this, is that there is what I call a complexity of progression. A first pass application is never taking on the most complex problems that a corporation is trying to solve. We're dealing with relatively simple data. What happens in the second pass? We start dealing with relatively complex data. The impedance mismatch problem: I'm not going to try to define it here, I think everybody here knows what it is. A search to rear its ugly head.

As people were running into that, using Smalltalk in the 90s, they started using something that we liked to call Multiuser Smalltalk or Persistence Smalltalk, which is Gemstone Smalltalk. Fundamentally this is an extension to the Smalltalk language that allows you transparent persistence. It's very, very simple. You just simply take your object, bind it into a name space, commit it, and it's there forever afterwards. The APIs are simple, programming is simple, you do it all in Smalltalk. The reason why I'm talking about this is because sometime ago Java happened. When Java happened a lot of the organizations kind of put the brakes on where they were with OO. "What's this new language? What's going on in here? Maybe we shouldn't invest so much in where we are. Let's put on the brakes and see where this is going".

I think now ten years into the progression of Java we are at a place where mission-critical, time-critical applications are being implemented and have been for the last couple of years. They've reached a complexity threshold where the OR impedance mismatch problems ere finally starting to show up. A lot of people who walked into going "Oh Java! It's OO! It's new! We can do lots here!" didn't realize at first that this was going to happen.

The long story short: You had expressed some interest, Bill, in languages that are extended with persistence. We've been doing that at Gemstone for practically 20 years. We know that it's possible, we know that it works. It's very simple, you don't have to have a relational database, you can use a relational database if you want to. My challenge to the team here, and to the people in the audience, is to start thinking in terms of transparent persistence as a part of the language. We've been doing that. But I think that it can be improved across the board.

Derek Henninger, Progress Software: Hello I'm Derek Henninger from Progress Software, and one of the things I've noticed looking into the introduction of this was that the last objects in database panel was quite a while ago. Someone looked at the one that occurred about 10 years ago. Everybody on the panel came from an object database vendor. Now I look up here and roughly half are talking about object relational mapping solutions and I think that is one of the trends that clearly has manifested itself since the last time this occurred and I think we'll continue to see that.

Progress is one of the vendors that has both an object database and an object relational mapping and caching solution. And I think we're going to continue to have progress and I think in the industry we'll continue to see the need for a interoperability between the object world and the relational world as well as other database formats. People are building their new applications in object orientated languages yet there is still a wealth of data in other structures beyond object databases.

That said, there are significant benefits that customers, that our customers get, and other customers and people on this panel get, from utilizing the power of an object database and not having that impedance mismatch.

Nonetheless though we see that interoperability is necessary. So even our customers who have standardized on ObjectStore for major parts of their applications still need to get data out of a relational database. They need that interoperability.

So I give a kind of an anecdote in terms of the benefits of a ODB: A recent customer of ours, Starwood Hotels, had an application built on Oracle. What they were getting for this very sophisticated query they were doing in order to measure room availability and stays and pricing and all that kind of that in one fell swoop. They were getting hundreds of transactions per minute. As they moved to an object database, they were getting hundreds of transactions per second - so roughly two orders of magnitude improvement.

For many of the applications out there I do believe that object representation can provide that kind of performance, particularly where performance and scalability and memory access is really critical.

Robert Greene, Versant: Hi, I'm Robert Greene, I'm from Versant Corporation. I've been working with Versant for about 10 years - also seen the ups and downs of the industry and I'm experiencing the winter, but I also feel that there's some potential for comeback.

I think that looking at the State of the Union, if we think about where we've been, where we're now, where we gonna go, it helps to put things in a little bit of perspective. Looking back we were very, very data-centric in our approach, focusing very much from a data perspective, driving that back up into sort of a structured programming paradigm and how does it operate over that data.

As things have evolved and we've begun to think more about objects and domain driven designs, we're naturally now at the stage where we are trying to figure out how does it take the evolution in the language layer and bring that down into what we're doing in the data tier. So the things you've seen happen in the Java space in particular with object relational mapping are first efforts to dealing with that problem. The reality is, most of the data out there is in relational form and we're increasing in XML form. So we need ways to deal with that but in the same time we need to appreciate that evolutions in the language layer are as important as abstractions that we put over existing data infrastructure. So that ultimately we can create solutions that will solve the problems at hand in a more efficient scalable way.

Erik Meijer, Microsoft: I'm Erik Meijer, I get paid by SQL Server and I work for Visual Basic and C#, so I experience objects and databases every day and the impedance mismatch.

But what I really want to talk about with you today is about car sales people. All think that you love them, right? You go and buy a car and then they say: "You can get your Corolla and a navigation system or you can get a Prius and your fuel efficient engine." But of course what you want is to have a Sienna *and* a navigation system *and* a fuel efficient engine. And the problem we have currently is that we also have this kind of disjunctive normal form for data. So it's like either you have objects and pointer based traversal or you have objects and foreign key/primary key based traversal.

So how can we get rid of this kind of disjunctive normal form in this ugly world? There are two ways to do that. One is to define a kind of an uber data model that some people try to do and then map everything into that uber data model. Or the other one is to try to unify, to look what is common between all these data models, how can we abstract over each model and over each collection type etc. And this is like where the future will be so I will leave that as cliffhanger for the next round to talk to you about that.

But before that I want to give this - let's say memorial candle. [Puts a candle onto the table] This is for all the companies that are still looking at the uber model and I'd like to symbolically burn this candle for them as a memorial that they will go down in peace. [Laughter]

Christof Wittig, db4objects: I totally agree with you Erik – no uber model.

My name is Christof Wittig. I'm the founding CEO of db4objects, the open source database company based in San Mateo. We started two years ago and we're certainly news in the space because we're the only object database company that ventured for the last 10 years. And there's reason to do that other than trying to get ourselves hurt.

In fact, more and more people - and that's the power of open source and the collaborative model behind it - more and more developers understand that relational database paradigms are not a solution for all persistence tasks in object orientated environments. And they just don't want to believe what Microsoft or Oracle often tell them: Use ORMs and they will make the pain go away. So our 15,000 registered developers and growing a thousand every month, they understand there are spaces where different persistence strategies are needed and one of the options is object databases.

Object relational mappers are a band aid for a problem, they're not the solution. Ted Neward called it very nicely the Vietnam of computer science: Object relational mappers are subject to diminishing marginal returns. You try to solve one problem and you get two new ones. You see it in added complexity and deteriorated performance. He says there are basically three strategies for object orientated persistence. Either embrace objects, object databases; abandon objects – that's a good strategy; or suffer from object relational mappers. Many developers, that for instance build applications for these cellphones [holds up a smartphone], or customers of ours like Ricoh and Seagate in consumer electronics, they understand that object relational mappers simply don't work on these devices. Market research shows that basically 50% of those developers write their own persistence solution - and that's in 2006.

So I want to say there is a case to be made for object databases and whenever you hear someone saying "Object databases have no use to object orientated developers", you just can tell them "You simply don't know the space enough".

Patrick Linskey, BEA: Well this should be interesting. I'm Patrick Linskey. I work for BEA for more than about a year; before that I worked for a company called Solarmetric, that makes an object relational mapping framework. [Laughter] BEA acquired us a year ago and I'm the lead of the team responsible for that and also getting involved in the rest of what BEA does with data. So I actually, from an ORM standpoint, get to solve two problems in one little kind of minute we have for talking here.

The gentleman on my left mentioned that object relational mapping is a band aid.

I think that this is a very common perception. There's a strong desire to find this uber model, the one thing that can solve all of our persistence needs, and one way to do all that. And I think that the most important thing to remember about objects in databases is that the format you want to store your data in, and the format you want to use your data in, are almost never the same. So there's inherently almost always some sort of mapping that happens even if it's just taking the bits in memory and writing them to be bits on disk. There's still a transformation onto disk that is happening and you're exporting, memories have different constraints for loading that data back in, etc.

So there's always some sort of a mapping happening, and the important thing is figuring out a solution that works for your business needs, your technical needs and that will work moving forward in the future. I think that the biggest thing there is the API that you use to interact with the data: The way that you access the data, and the way you manage your transactions, and the way that you manage your units of work, and things like that. The mapping exercise, storing it into whatever format you might be using, ideally, if you can use a unified API for communicating with your data or some small number of unified APIs that are interoperable, then how you map your data becomes a tuning exercise.

Whether you're storing data, whether you're using an object database or a relational database or a LDAP store or a network database or a flat file or whatever, it becomes an exercise in politics and optimization. Rather than in kind of upfront, let's start designing, now we must decide how we are going to store our database.

Craig Russell, Sun Microsystems: Hi, I'm Craig Russell, I'm with Sun Microsystems. I came to Sun in 1999 in order to do object to data representation and I don't use the word object relational data. From my perspective the important thing is how the programmers use the data and what they need. The less important thing is whether it's an object relational mapping or a flat file or a XML file. I like to concentrate on "How can we make the persistent data appear to the programmer already made for active use".

So unlike a used car salesman, which I really don't identify with at all, I'd like to use the analogy of "Trying to get a nice meal at a restaurant". The database is behind the doors in the kitchen and we really have no idea what goes on back there. The health department does, but certainly the diners don't ever see that. What they see is a presentation in front of them like objects such as the dishes, the flatware, the silverware, the goblets and so forth.

I really have to say that I think "impedance mismatch" is completely overrated. I'll give you an example of impedance mismatch that might work: You sit down to your table and you're given a goblet. That's what you have to eat with. Out comes the chicken and you try to eat the chicken with the goblet. It already has got some wine in it and you add some water to the goblet, and the peas and the carrots, they're all in the goblet.

That's a real impedance mismatch from my perspective! The way you like to eat your meal is, it comes out of the kitchen, it's all delivered for you, everything is in its proper place. You've got the wine, you've got the water, the various things, you can eat it with a fork and a knife and it's all just really easy to use. Easy to use food, that's my objective in this talk here.

So you can think of an object database as the dishes prepared in the kitchen: It comes out ready for use. You look at it, you see it, you eat it. You can look at JDBC as the food comes out in a big platter and it's your job to serve yourself, family style basically. You take a little bit of this and a little bit of this and you put it on your plate. So you construct the objects that you're looking for. And you can talk about an object relational mapper as the staging area in between the kitchen and where it's actually going to be served. The waiters are doing a tableside presentation, they mix all the stuff together, put it on the plate and serve it for you. You never had to do any of the work, you saw it done. Maybe a look behind it, you saw it done, you know what is going on. You know the data coming out of the kitchen is not in the form that you're going to eat it.

I'll just leave you with that, in saying that there's a real separation in my mind between what is in the kitchen and what shows up on your plate. My focus is what is on the plate.

William Cook, Moderator: All right, thank you. If anybody wants to paint the picture of where we are going to be in 10 years, I'm going to give you a minute to do that.

Bob Walker, Gemstone Systems: What we found since we're already doing transparent access to objects without having to do anything other than to say "Wait, or I want an egg" and the egg is still there. I think that problem has been solved in terms of OR mapping or impedance mismatch, I like your analogy, I think that is very good. I think the next step and from what I'm hearing from programmers, I hear things like "I just want my objects. I just want them here and I want them now and I don't want to have to mess with it, I just want them there, I don't want to have to deal with the database, I want all that stuff taken care of for me."

I think the next step, and I think we're seeing this at Gemstone, is what basically is a distributed in-memory live object cache that has transactional attributes but it doesn't deal with disk space storage whatsoever. I think, five years from now, we are going to memory cheap enough and fast enough that there won't be any discs. There will simply be in memory objects, ubiquitous throughout the enterprise, a robust sea of objects always available for the programmer. They don't have to mess with OR mapping, they don't have to mess with object databases, they don't have to mess with SQL, the objects are just there ready for the taking and for the use.

William Cook, Moderator: Great so we have a convention. If you agree with that, you can just say "Gush!" I completely agree with everything and I don't need to say it again. So if anybody wants to agree with that they can and if anybody give a different view…

Derek Henninger, Progress Software: So I'll "gush" - but I think "Fine there is this distributed object caches in memory, great"…each object application, even two applications of the same domain, often have different object structures. There's going to be transformation required between that. I completely agree there's significant transactional characteristics that are going to be associated with that, that need to be managed and I think it's our responsibility to satisfy that and the needs of the programmer and to make that simple and easy to use.

Beyond that, I still think that the relational model and other data models are going to exist for 10 years. We've been saying that some of these mainframe database are going to die and they still sure as heck haven't. That kind of mapping and transformation into the objects in memory representation is going to need to occur and you're going to need to do it in a way that does not impact those legacy applications that nobody wants to touch. Those COBOL programmers that we talked about so affectionately earlier, those guys don't want to touch their applications. You do something that affects their application logic, and keep in mind those old COBOL applications make a lot of assumptions about what the transactional characteristics are of the data they are managing. So there's a lot more complexity to it and I think the mapping is a key part of that.

Robert Greene, Versant: I've been stuck with that recurring dream myself but I would tend here to side with Erik. You're not going to get away from the realities of having to worry about the transaction nature of your logic and also the raw scale. When you start talking about dealing with information which is in the terabyte and even approaching the petabyte range you're gonna be hard pressed to blindly deal with that it's that there are just objects that are available. You're going to have to have new constructs that are either built into the mapping and transformation layers or natively in the language, that can help you as a programmer to deal with those things and as transparent way as possible, but I think it'll never be completely transparent – at least not in my lifetime.

Erik Meijer, Microsoft: I think it's a delusion to ever think that all your data will be available as objects. Over time there will be more and more data models and forms of data: MP3s, audio, video, whatever. And you want to deal with all these forms of data, you don't want to get stuck. So the only solution is to look for something that very much rises over all data models. And there is a solution for that, it's called Monads. This was invented by the mathematicians in the 1920s.

But if we look ten years from now: I'm going to make a little digression here because I was judging the students competition and none of the students told me "My work is impractical and maybe in a hundred years there will be a use". But if you look at this theory of Monads that was invented by mathematicians in 1920 or something: They had no clue what this was useful for, so my belief is that in ten years' time we will be in a crisis because the current generation of students are only into impractical stuff and they're not generating the theory that is necessary to produce the kind of next generation technology. It's like oil, we're running out of mathematics quickly. This is my sober view of from ten years now – it's sad.

Christof Wittig, db4objects: My prediction of what is going to happen in ten years is sort of a cultural war who owns the data. We'll decide in one way or the other – I don't want to make a prediction in which way. Actually, the fact is that in enterprise applications the data belongs to the DBAs and not to the developers. I think that it's time that developers reclaim their territory. That's why we have so much traction in the embedded space which is also dubbed as zero administration – lack of DBAs. But we should think a little further and take ownership of the data persistence. And that's where I disagree with Craig. It does make a difference whether your meal comes in ten minutes or ten hours - a developer should care about this. Even if the API is the same - he has a menu and a bill to pay in the end and a meal in between - it does make a difference whether it takes ten minutes or ten hours.

Also the quality: Can I properly persist inheritance, can I properly really persist the power of object orientated languages. The fact is that these kitchens that we have to date don't allow and cater to those needs. So we have to reclaim those territories and put ourselves in charge.
What works well in the embedded space, I think will very well be picked up with service orientated architecture in the enterprise space where the application, the database form a contained silo and no MIS consultant will ever touch the database but only use the predefined service gateways which are ownership of the developers themselves.

Patrick Linskey, BEA: I guess to that I'll say "Gush". [Laughter]

One of the things that Eric said is really interesting. I think you'll see a lot of different types of data models coming out. I think the next one we're going to see in the next five or ten years is this kind of this concept that we were talking about earlier, of data grids and data in the network, like data on the cloud rather than data on the little disk.

All this different new modes of data storage and data durability are going to end up demanding different APIs and different ways of interacting with them. That is definitely going to make things more interesting. But I think that also ten years from now we're going to use a whole set of new languages. No one is going to use Java, no one is going to use C#, everyone is going to use Dflat and Cava or whatever. We always do this: We throw away all the knowledge we learned and start from the scratch again. And I think ten years from now we'll be going to do that again. We'll have the same discussions about what's the right way to do it, what's the right way for the IDs and what's the best storage. We'll probably disregard the past. That's not what I hope happens but it's what I think will happen.

Craig Russell, Sun Microsystems: I'll just say that I really "gush" about the fact that the big crisis coming is about who owns the data. Is it going to be the people who produce the content or is it going to be the people who distribute the content? The digital rights management, I think we haven't seen the third act or the fourth act play yet and that's going to be somewhat critical. What I do know is that the data, you know who owns it, the corporations, and the people who pay the piper which are the corporations, there'll be a significant number of those who demand very fast access to the data and they'll hire programmers who can deliver that fast access.

I'm sorry because in my analogy I didn't carefully enough recognize these staff and the people in the kitchen in serving this meal. I do recognize that a good presentation is a well-orchestrated operation involving a whole series of people from the backend, from the people who actually order the meals, order the raw materials and prepare the meals to the people in the back office accounting-wise who magically transport your plastic and deliver you the key to the exit door because you don't leave without you pay.

So there's a lot of roles going on in delivering persistent data. I'm describing what my focus was: To deliver the ultimate dining experience. I recognize that there's some on the panel here who are more involved in the back office kinds of operations.

My vision for the future is ubiquitous data, but the data is as distributed as varied and the ownership varies with the data. To the point of all accesses aren't in this particular style, the needs of the cellphone or the PDA are very different - I completely agree with that - and I don't want to eat my chicken cordon bleu with a cellphone.

William Cook, Moderator: I've been a programmer for a long time and I'm not sure I trust myself with the data - so it's a little scary.

We have a couple of questions [from the audience] here.

Basically there's a question: "What is happening with ODJ?" I just want to elaborate on this a little more, this is the latest in the long line of standards for accessing relational databases starting with ODBC and we've got to have more acronyms and standards and proposal in that spaced than in any other space I know. There's something clearly going on here, we're having a new proposal every year or two, right? How many of these: JDO, EJB-this EJB-that, the dot.net versions of this stuff, what's going on here? Is this ever going to end? It's roughly the question I have so if the people here involved in that can answer that.

Robert Greene, Versant: I just have one question to your statement which is that you're not comfortable with the data but are you comfortable with your models because I think that goes back to what Christof was talking about, taking control again of the models and thinking from a domain driven perspective and not having to worry about what's going on with the data, how it's stored is part of that model, but: Be comfortable with your model.

William Cook, Moderator: That sounds good to me.

Patrick Linskey, BEA: Regarding the question of the audience I think that there are a lot of acronyms out there and in the defense of the persistence role: The web services world has even more.

William Cook, Moderator: They're not replacing their previous ones; they are adding more of them. But in the relational world they are always replacing the previous one.

Patrick Linskey, BEA: That's true; we do like to reinvent the wheel in the OR mapping world. I really hope that the standards in the OR mapping world settle down over time, because I think that the key thing, a good OR mapping standard, and a good object database API and a good LDAP API for storing objects in an LDAP server all end up looking a lot the same. You can do conversions between the different APIs with an easy "sed" script. So it's in the best interest of everybody, except for the vendors, for there to be strong and adopted standards for in whatever form they get delivered for how you access data as objects. Because this is a problem that a lot of different people solve. Here's my way of accessing data as objects and the users of all these products are the ones that benefit from having a common API on top of them. Having extensions to APIs is fine but having some basic subset is really something that, any vendor that says something is not deliverable is trying to lock you into their product - and that is not nice.

Christof Wittig, db4objects: I wanted to speak to one of the initiatives of William Cook himself, you may not know that he is the author of the safe queries concept, among others. We have implemented that in our database as native queries and basically it makes the program language be Java or dot.net itself the query language. It's totally native, it's type safe, it's object oriented, and it's fully optimized. That is one way to reclaim the territory, we don't speak the other language of the DBA guys - SQL, was written for end users not for programmers. We use our language because we're better in that. No one can be perfect in two languages at the same time – no matter how good you are. So one standard is: Let's take our *language* standards and just use it, for querying for instance and for persistence tasks. So, no new standard please.

Bob Walker, Gemstone Systems: I want to second what Christof just said. I think having to deal with two different languages is part of the core of the difference that we run into in mapping objects into relational data. Our effort at Gemstone has to be try and keep it in a single language. So your queries are Java, your queries are Smalltalk. There's been an effort, ODMG 3.0 with the definition of OQL to define a standard object query language. This is something that would behove all of us as vendors to take a look at and participate.

William Cook, Moderator: I want to get to you object database people, any other comments whether the acronyms soup is going to ever end?

Derek Henninger, Progress Software: I completely agree that having a language be the access mechanism is right and an object store certainly supports that strongly, but again: If you need to get information from another source that's where it becomes impractical and you do need a standard approach for doing that in order to interact with other solutions.

I would love to see a standard. Personally after following it for ten years I've gotten tired and I'm glad Patrick is working on it because I burnt out on that, good luck!

Erik Meijer, Microsoft: Because we're running out of three letter acronyms we're introducing four letter acronyms, now with LINQ - so we have some more space to grow the alphabet soup.

Craig Russell, Sun Microsystems: I think that there'll be programming languages created at pretty much the same rate as we've done it so far until we've decided that there's no need for another language. Some languages will become obsolete. But I don't see any a priori decision not to create other programming languages.

William Cook, Moderator: The rate of programming language introduction is a little slower than the object relational mappers.

I have some other questions [from the audience]. This is a question for the object database people.:

"If the object winter is starting to thaw, what happened or what changed to make this happen? What is different?"

There are some comments here about all these issues of transactions and concurrent access to data and multiple representations and all the things that have heard about…the problems which were reported were either true or not about object databases. Is there anything different now?

Robert Greene, Versant: We've kind of moved past that initial phase of how we put pretty pictures on a web page in Java. We start to think more seriously about solving our server side with serious scalability problems with that same language. With that, I think people really at large embraced object orientated programming. Smalltalk started the revolution, nearly died initially with Java in some aspects and grew again and have people thinking again truly in ways that they haven't before.

If you get a crowd of a hundred developers out of the university now and find out about how many of them know about OO, and know it reasonably well and understand the concepts, it's probably all hundred if not 95. If you went back prior to Java and did the same thing it'd probably be about 50:50 at that point, so things have changed quite a bit. So with that people again are starting to focus on how it is to solve very, very complex problems of all themselves and they're thinking in terms of modeling problem domains and creating models which represent the solution to the thing they're trying to solve. And with that they're only now coming back to the point where there is this thing in the background, which is the way that we've always represented our data and how it is that it imposes these restrictions upon to what we're trying to do with our models.

And that's I think precipitated some frustration that had been absent for a long time and so I think that the attitudes and the problems that are trying to be solved are precipitating a change in that direction to re-evaluate what is it that we're doing there, that makes it so hard. Somebody who doesn't even know how to program can go to Second Life and they can design a three dimensional space and they're not even a software developer. And yet we haven't figured a way to distract ourselves away far enough from the way we deal with data yet to make that not painful for ourselves. That has just got to stop.

Bob Walker, Gemstone Systems: I agree entirely, that was very well said, thank you.

Patrick Linskey, BEA: I have a question for the audience: Raise your hand if you've ever written an object persistence framework. [About 40% of the audience show their hands.] Now raise your hand if you're still using it. [About 15% show their hands.]

William Cook, Moderator: What do you conclude?

Patrick Linskey, BEA: Developers like to write object persistence frameworks. [Laughter]

Bob Walker, Gemstone Systems: Developers like to keep things complicated.

Christof Wittig, db4objects: I think a big game changer is open source. We're not forced to eat what Microsoft and Oracle spoon-feeds us anymore. [Laughter, clapping]

Erik Meijer, Microsoft: [Smiling] I'm not even going to comment on that, I'm above these things.

But the thing is that I want to go back to one of the earlier remarks that was made. I think there's a big difference in the way you store your data and the way you access it. I think especially objects are a very bad way to store data because the pointers go from the inside of the objects to the outside. That means that you can never have different relationships between objects without modifying the objects.

Here I'm schizophrenic, I think there the relational model is much better or something like RDF where you can have pointers from the outside or you can kind of take read only data and then kind of connect them, put relationships between them. I don't see how to do that with objects even if you have dynamically typed language, it would be still kind of modifying the objects to form the relationships.

Craig Russell, Sun Microsystems: I like to comment on that: The form of the data that you get depends on where you are using it. In memory, for small connected devices with 2 gigs, 5 gigs, 100 gigs, what are we talking…in 5-10 years…a hundred gigs that you are going to carry around with you? Your entire collection of favorite movies? The relationships that you can navigate in memory are very different from the ones you navigate if you happen to be Paramount Pictures and want every scene cross-annotated, that's a much bigger problem. But if you want to just do some query on your cached information there's no point in storing it in tables.

William Cook, Moderator: Let's change the focus a little bit.

We had dinner last night and I asked some gentle questions to get to know the people a little better. One of the things that kept coming up is that the issues we're really ought to be talking about are not technical issues perhaps entirely, but our business and social issues, so we have a couple of questions along those lines.

"How can you pursue object orientated databases when corporate customers insist their data be in a relational database?" and "We have these databases - what are the problems we see in the industry?" Because what is really happening is all this is being played out on the desktops of DBAs and programmers in the industry.

What's the social and business climate, how is that affecting you guys?

Derek Henninger, Progress Software: I want to relate to that, a comment earlier was that we as programmers need to fight back. I think our chance to do that was the dot com boom, at least what we saw that was when DBAs had less power and anything to get that new app on the internet was acceptable. What I see is the pendulum has swung back to the DBA and the core CIO and all that infrastructure they built.

I believe that the relational world is going to continue for a while, I also believe strongly in the value and benefits of object databases therefore I see no alternative but improve mapping and the transformation tools around that need to become a lot more sophisticated. It's not just simply OR mapping any more. I think it's object to object mapping as well. As you look at two different applications maybe written with a Versant database and an ObjectStore database and a Gemstone database or a db4o database. At some point we're going to want to map between them. So I do see that's going to be an important technology advance in order to make all this practicable, not to mention relational and hierarchical and all these other databases**.**

Patrick Linskey, BEA: I think I agree a lot. One of the things that OR mapping does a really good job of, is organizationally, lets people get along in the organization rather than kind of negotiation 101 is that when you sit down on the negotiation table, you're not trying to win; you're trying to come to a solution that everyone can agree to. That's where I think OR mapping does a pretty good job of, in a lot of companies where you have DBAs say "Thou shall use that schema", if you want to add something put it into the top of the pipe over here and then two years later we will tell you whether or not we will accept your change.

What I've seen OR mapping do over the years is, companies say: "We can do that it's just "going to cost us this much to set up this mapping" or "It's going to be a little bit slower" or "We need to write a little bit of custom Java code" or whatever the case may be. It puts the developers and the DBAs in a position where instead of using political muscle to create a winner in a given situation you actually get to blame a vendor and work together with your DBAs and developers and say "What is it what we want to be able to do? What's the data model, what's the object model, how do we make them bridge?". Then you go around and you see the different people who are pitching you on software. How can you make this happen? I think that's essentially why OR mapping is popular, because there are so many people who must use their schema that is used by 70 applications over the course of the last 15 years. They can't change it, so but they want it to behave differently, they don't want to fight that battle any more.

Robert Greene, Versant: How many of you have seen the size of an object relational mapping layer? It's pretty enormous, and that has to execute on a CPU and that's a lot of instructions and there's just no getting around that, the cost involved in using that approach. One of the things that is going to be championed up into the upper management in some of these companies is the cost of computing. If you look at the message that is coming out of companies like Sun, when they start talking about the fact that it's going to cost you more to run your computer for the life of that computer than it is to buy it, then you need to start looking at things like when I choose to use a relational database for this application, if it means that I need to run on twice or three times as many CPUs as I would be running if I chose a different technology, is that the right decision we're making?

If you look at some of these applications out there, I just happen to be working a lot in the massive multiplayer online gaming space of our friends back to Second Life again, but they're talking about 10 million users and 150,000 servers by 2008.

So imagine if choosing one technology over another at the database layer meant that you could use 70,000 servers instead of the 150,000 servers. That's the kind of thing I think that IT management -- if they're good at what they do -- ought to be looking at when making those decision instead of just "No my guys know Oracle and that's it". So we're an Oracle shop we're going to use it blindly without a proper analysis.

Bob Walker, Gemstone Systems: I think that's really true. And one of things you should keep an eye on is "Where's the money? Where's it going?" There's a faction in industry that is well vested in selling more hardware, networks – at odds with optimizing the software.

William Cook, Moderator: I want to come back to the social question.

Bob Walker, Gemstone Systems: Sure. Basically it's a matter of being able to take risk or being risk averse. It's like the old axiom nobody ever got hired or fired for using Big Blue. Nobody ever got fired for using Oracle or DB2. I think that there are definite social issues and they have to do with risk aversion. Sometimes it's a hard sell to get people even look at what available object transparency might look like to them.

William Cook, Moderator: Here's an interesting question: "What about the agility mismatch?", "Current tools allow me to easily refactor my class structure but not my schema".

Christof Wittig, db4objects: Exactly! That ought to be addressed and that's why db4objects for instance supports an automatic schema evolution. You can basically just adapt it and you're in charge. There's no dual schema setup. This is the silver bullet not for every application but for many it is. And if people fight back and said "Object databases are all crap", no it's not. It's a matter of context. And those people who disclaim that or try to tell you "Object databases are no use" - yes, they are there. They are mainstream, they are freely available under the GPL, and they work. Try them and you find that it's so much easier, you can go home at five. [Laughter]

Robert Greene, Versant: One of the applications that is running on Versant, I thought that I was going to stay away from this, but it's a War Games 2000 so it's a global battle management simulation that has been running for a long time. Every single run that they do involves the schema in some form. And they must allow all the clients to work with the data as well as new clients. And so those capabilities were built into objects databases, some of them better than others. But those are important things that we all need to look at moving forward in solving how we deal with data in the language and the way we use languages.

Erik Meijer, Microsoft: I think data should not evolve. You want to present it differently but I don't want certainly all my mp3s on my hard disk to change and now half of the applications don't understand the new format. So I really think it's a bad idea if it's easy to change your data. I don't buy your argument at all, sorry.

Patrick Linskey, BEA: I would disagree with that. I would say that models should stabilize over time. As they do so solidifies the data that they represent. But to say that you'll get the model right the first time: it just rarely happens.

Bob Walker, Gemstone Systems: Some business requirements change, new forms of businesses are created. In insurance there are new rules let's say, in other mathematical forms, there are new computations, data is going to change – the structure of the data is going to change. I can see an mp3, I might want to have the format the same but if you're writing a new insurance policy or you're creating some new stock portfolio you may have different characteristics of your data than what you originally had.

Craig Russell, Sun Microsystems: Data is going to change – I guarantee it. And so is the schema, and so are the government regulations, and so are the opportunities that some company has to add a little bit of information and gain a slight competitive advantage. This like Darwin at work here. So data will change. And if you think your schema and data won't change you're literally going to be living in the past – don't make that mistake.

Patrick Linskey, BEA: I think the Ruby on Rails guys have done some really interesting work with their schema in data evolution strategy, essentially modelling after the kind of a combination letting you continue to use the underlying capabilities in data store rather then putting a new semantic on top of that. And also using a kind of a style mechanism, of protocols, of little scripts to upgrade and downgrade between different versions. That's a compelling way to allow developers to mutate their schema and model over time.

One of the really pesky things that it gets around is: essentially you need to have some declarative mechanism of doing this and you can't do this in a Java programming language or in a strongly typed programming language very easily. There's this weird thing: If you change your schema in an incompatible way, which classes are you executing when you make that change? The moment of the change happens both different sets of classes must be understood, it must be available. It's an interesting challenge, I think the Ruby on Rails guys have done it – at least from a relational standpoint – in the most pragmatic way I've seen yet.

Craig Russell, Sun Microsystems: I think dynamic languages are a way out of that and in fact I'd go as far as to say: As much as I love Java, static typing and fixed typing is way overrated in this universe. I'd pose it as a challenge to the audience to say "Why isn't change part of your model? You know it's going to happen – build it into your model!"

Bob Walker, Gemstone Systems: We've had really hard experience between both Smalltalk and Java, a dynamic language and a static strongly type language in the realms of schema migration for object schemas. And you're absolutely right. Doing it in Java is really difficult. Smalltalk lends itself so incredibly well to that kind of dynamic schema modification and multiple class versions. It has been our experience that Smalltalk is a wonderful language for this sort of thing and Java is pretty tough. I agree with you, I underscore that, dynamic languages are really something worth considering.

Erik Meijer, Microsoft: So maybe here my Haskal background can show through. I'm not against changing the schema or the model of the data, but the data itself should be read-only and you should not modify your data, so it's very well possible. There might be other people looking at the data so you should not modify it – destructive update is really bad. That has nothing to do with dynamic typing or static typing or evolution or anything. I just want to make sure that, before you say I'm a dinosaur…I'm the one that's declarative here.

William Cook, Moderator: We're really starting to get two different viewpoints here, the sort of object relational where the data is owned by someone else and maps it versus the object database, that's the right presentation and everything is just seamless and you don't have to do a lot, but it's all handled in the sort of object style. Here are some questions related to that:

"Tell me a compelling example that illustrates the advantages of an object oriented database over a relational database?" Just give a specific example and, if you an object relational mapping person, you can explain that why there isn't a compelling reason for that.

Robert Greene, Versant: There are 70,000 servers versus 150,000.

So that's just one example and I'm sure that Christof can give another example.

Patrick Linskey, BEA: As an OR mapping person I think a thing that a lot of object databases tend to do better than the relational databases right now is data federation, is partitioning data over a large number of servers. A fairly compelling example for this in America was the telephone exchange systems where you knew that if you had a 202 number, that went to the DC exchange and if you had a 415 number that went to the San Francisco exchange etc. So data models that are inherently partitionable and that's changing because we can port our numbers now and everybody has cellphones. So those types of systems are things where relational databases tend not to scale past a single machine and DB2 or IBM and Oracle were working hard on this problem. This is a problem people are trying to solve. I should point out, that it has little to do with the underlying model but a lot of times when we look at things that are better or worse than one model it's not actually because of the inherent academic properties of the model but often it is because of different implementation strategies and assumptions that the companies developing the products have ended up developing into their software products

Christof Wittig, db4objects: I want to stick to this specifically so I just give an example. A consumer appliance which is shipped in the 10,000s and margins are razor-thin, so every single CPU circuit really counts. Most of our developers really look in either writing their own persistence solution or using serialization or flat file. Think of an object database as seamless as serialization but it doesn't break – it's as seamless. And that's when they choose db4o, so it's much more robust. And as a result, their customer service director said: "The number of service calls has gone down 10% since we use db4o instead of object serialization.

Derek Henninger, Progress Software: So a couple of examples from ObjectStore, in their case it's a very complex indexing structure that the relational equivalent was something like a 20 table join. You can do that kind of thing in an object database you have a lot more control over the sophistication of the structures that you're going to build, you're not limited to rows and columns.

Another example would be GIS information which is really much better suited for an object database because it has a kind of navigational feel to it. You need that kind of navigational approach to things in many applications and object databases are far superior for that, you aren’t doing the database foreign joins key lookups in order to get the information. Although it’s a power of a relational database it’s not something that’s done efficiently.

Bob Walker, Gemstone Systems: I’m sure that all of us have concrete examples of where traversing a persistence model by reference rather than accessing by query outshines a relational database. We have one at Gemstone that I can talk about, I can’t talk about their name, but a large telecommunication company does all their DSL provisioning with their objects stored and code running on Gemstone VMs. They found that they couldn’t do this with the off the shelf product in its traditional accessing of DB2. They simply wrote to the Java APIs and started storing that complex DSL network data in our database. They can provision a DSL switch in minutes or less. It’s one of their key successes where the competition in the industry is a day, two day, twelve hours if you’re lucky. I think there are a lot of examples where on a certain level of complexity the object databases really start to outshine relational databases.

I have to say that I’m not opposed to OR mapping. I also completely agree that there’s a lot of relational data out there that needs to be dealt with. What I’m trying to look at is “Where are we going next? What are we going to do with all this stuff? How are we going to access in a more homogeneous fashion throughout in the enterprise?”.

Patrick Linskey, BEA I think there are relatively small numbers of applications where one particular data model is superior to another. I’ve seen more examples of situations where an application runs faster for this section and so of that section, always is fast enough or always has enough features or whatever. More often or not the appropriate running data storage solution is probably going to be good enough for the vast majority of any needs regardless of which different kind or varying technology it is. It’s really only in the kind of corner cases that a given type of storage pattern makes the difference for the application in the here and now.

Craig Russell, Sun Microsystems: Just one more comment to emphasize that a lot of database people think in terms of third, fourth and fifth normal forms where there’s exactly one copy of each piece of data. It’s been my experience that the successful applications deal with multiple copies in various stages of completeness and coherence. In order to get an application that works really well you need to be very aware of the transient nature of the data, getting more ossified the further back into the system it gets until you finally actually do buy that on the internet. Once you’ve made the purchase it should be firm. You shouldn’t have any ambiguity or any question.

Robert Greene, Versant: I just want to comment to make sure that the audience realizes, that at last from my perspective, query versus navigational access, which is often sort of focal point of discussions when it comes to object versus relational databases. It’s one of the things that certainly I learned in working over 10 years with an object database company, that those are complementary things that you cannot do in a holistic, pure, navigational based approach.It may work for something like db4o in the embedded space but it doesn’t work in a large scale enterprise application. You have got to have complementary technology from that perspective. You need to go out and query the 80 million instances and bring back some set of objects which are the starting point for your use cases with which you then use navigation to drill down into and exercise your business logic in your model.

But you've really got to have both of those things and object databases have evolved substantially in that area of query as a compliment to the navigational capabilities.

William Cook, Moderator: This is a bigger picture question that the people were asking: "What is the impact of service oriented architectures? How is that going to affect the object persistence?" But perhaps the question here is: "It moves the data further away from the application. Has it a tendency to do that?" And one person was asking along those lines "What about data federation and federated queries? Are we going to have an object relational mapper that works over service oriented architecture and over a federated data?"

Patrick Linskey, BEA: I think that those two questions are very, very related. A really interesting thing that Erik mentioned earlier is that, in the service oriented world, I think that the myth of SOA is that developers only coded what this app, everything is beautiful and you use lots of brackets. I think that you still write your services and a number of services are still written in Java or C#. And those services are the things that talk to the database, so then you combine a different bunch of services and the trick is really: a) how do you avoid too fine-grained services? They are so fine grained that the application performance just plummets, everything is just XML and b) when that does happen or whenever we can optimize around it how do you recognize that two different services are talking to the same data store and make optimizations of how to query across them? How do you do this problem of federating a query, taking a single query that spans most of the different data storage solutions, maybe objects, maybe relational databases, maybe LDAP, maybe XML, maybe just services directly, maybe old legacy CICS mainframes and puts that together into something approaching the optimal query path to get all the data for the parts of the puzzle. Sadly, I'm not the best person for b) to talk about that but I know that there is something that is a big focus of one of our team down in San Jose and it's really, really interesting stuff. I think that the kind of service aggregation is really fascinating and piercing the XML barrier is something I'm really interested in. How do you make these services less opaque to the environment they're running in.

William Cook, Moderator: I almost wondered whether there isn't a service oriented impedance mismatch starting to arise. That's somehow related to the data one but the services are over there and how do we talk to them?

Patrick Linskey, BEA: How do we pretend that they're all somewhere nearby?

Erik Meijer, Microsoft: So one thing is – remember that candle there – I think the service orientation is one place where people have fallen into the trap of having this kind of unified data model that they think XML will solve everything, but I am personally not such a big believer in XML. Hardly anybody needs comments, PIs, significant, insignificant white space, name space, all the crap that XML gives you. I think a lot of people just want to have their own data model so again that's why please for may having a solution that works for any data model - don't fix yourself.

The other thing is that you want your data to be read only, you don't want that, people probably cannot modify or update data, the data will be distributed etc. So I think this all points to Monads again, to bring that up again. It works for any form of data and it focuses on "What are the common operations on the data?". Such that and then you can distribute these operations to the data or you can bring the data to the operations. But it becomes just a matter of optimization but you have kind of unified, really parametrized for all data models instead of picking XML or whatever will be the next big hype.

Derek Henninger, Progress Software: One of the things that we see at Progress is - actually a lot of the companies that are using service oriented architectures, the loosely coupled, message based, are often using it to integrate two different databases. XML becomes that common model for a specific application, the other where I guess we'll see that is where you need to communicate across different companies within an industry. Telecommunications is a great example of that: Your call gets transferred to a bunch of different providers, they need to communicate and they have common models that they are trying to define as an industry. It's a pain, it may not be successful but it is a way that they are interoperating. Basically every major industry is coming up with their own standard and I think that the solutions need to integrate into that.

One of the challenges that you look at service oriented architectures and in particular as you build business processes, kind of the silver bullet, where everybody points to is going to have all these services that everyone has built and then you can build this wonderful business process on top of it. One of the challenges that happens, is you work through from the first step to the second step to the third step is the data kind of builds up. We call it the snow plow, so as you go through each stage of the process in order to ensure that this next process and the next service gets the information it needs to proceed properly, you're accumulating all through that. And one of the things we've been talking about at Progress is whether there is an equivalent at the service layer, is there an equivalent sort of data channel on the service where you're keeping the individual processes up to date with the latest data so they can process the latest requests as opposed to having everything just built up and your XML file gets bigger and bigger and as the process gets more complicated the XML file continues to grow.

Christof Wittig, db4objects: I think it's a question that's very related to object database. If you look at a company like salesforce.com, to the outside it only has service APIs and many companies have shown it very well integrates into their business processes, and you don't ever access the database on a raw level. So I think people underestimate the change in the way of how ownership is redistributed through service oriented architecture. It actually gives you the chance to slice away applications and outsource them. As a result the application developer gets back into charge of how the data is stored, how it's most efficient and most performant, rather than having the DBA with his data store that sits on top of all the applications in my enterprise, you have a silo. And the application developer now is in charge of how to persist data in the most convenient way, which caters to the needs of service oriented APIs and gateways. So that changes a lot and that opens the door for using object databases in an *embedded* way if you like. Because the end user is not aware... do you know whether salesforce.com is running an object database or an oracle database? And why would you care?

Erik Meijer, Microsoft: So did something in your youth happen, some dramatic experience with a DBA or something? [Laughter]

William Cook, Moderator: I wanted to comment on that too. Besides the impedance mismatch at the technology level I certainly have experienced a lot of impedance mismatch with database people versus programming language people, there's a real cultural gap there. I think that historically we've had a big problem where the application people actually don't know a whole heck of a lot about data and have proposed to have done a whole lot of things that aren't really to take advantage of the knowledge.

And there's a similar thing on the data side. They don't necessarily have an understanding of all the issues of modularity so we haven't really gotten to the point where we are talking to each other efficiently on sort of a cultural level. Maybe I'll throw that out you can comment on it but we only have time for another question I think and I wanted to pull together. I actually have two here that I could go for.

One is: "I've heard this word model a couple of times and I've got a question about it.
When will we see the HTML of data models?" I wanted to recast the question, there's a couple of other ones that have hinted at it. Which is "Maybe the service oriented architecture is going to have *one* impact but maybe will even have a bigger impact on this whole model driven architecture?". Is model driven development something that you guys want to talk about? And is that some way where we can get some more common ground?

Bob Walker, Gemstone Systems: I think you mentioned a couple of different terms here. One of the things you just described was the difficulty of communication between the programmer camp and the DBA organization. I think that goes all [the way] from the customer level. The end user is defining the application all the way to the architects down to the programmers down to the data. It's pervasive in an organization.

This being OOPSLA I'm sure people understand that one of the ideas behind object oriented programming was to build a common language that could be communicated on evenly across the border in terms of business objects. Those are abstractions that will get brought down and reduced into implementation objects but I think the object oriented analysis and design techniques are what's needed to foster that conversation.

I think, given the way the increased demand on people's time has happened over the last decade, a lot of this has fallen by the wayside. And so we've fallen into a space where we are not communicating any more.

Erik Meijer, Microsoft: I must say that it's a little bit sad that the panel answered the question on model driven development because if you look at the object people here, they all talk about modeling, objects model, real world things. If you talk to database people, they talk about their tables, model real world things, customers, orders and whatever so we've been always doing a kind of model based development. Every new generation has its own interpretation of what the model is. So I think there's nothing new here, nothing has changed. You can give it new fancy names but it doesn't solve any new problems, it just gives different names to the same things and it only adds to the confusion. It creates yet another impedance mismatch.

Bob Walker, Gemstone Systems: Actually I was talking more about building a common vocabulary than a vocabulary that causes divisions and confusion.

Derek Henninger, Progress Software: Back to the previous question. You start to build models, you're going to have greater need for additional models and being able to communicate between them. Certainly, the big massive model is an impossible problem. On the other hand if you need to integrate a bunch of different sources and have them maintain consistency you need some way of abstracting that. Your DBA is not going to do it, your programmer is not going to do it in Java or XML. You need some way of abstracting that and having tools help to identify the inconsistencies.

William Cook, Moderator: I think we are out of time so I want to thank everybody for coming, thanks for the speakers and the panelists.

Thank you very much!