# Making Sense Out of SQL Server on Linux

Hakan Jakobsson
hakan@hakan-jakobsson.com

Version 2016-04-14

## Introduction

Microsoft recently announced that it will port its SQL Server database product to Linux. A few years ago, such an announcement would have been completely unthinkable, and even now, there are probably quite a few people who are shaking their heads for good reasons. There are good arguments for why it will be difficult for such a port to get any traction.

The announcement of the Linux port "happened" to coincide with an aggressive SQL Server marketing campaign targeting Oracle that tries to entice existing Oracle customers to migrate, not to the long-in the-future Linux port, but to migrate here and now with promises of better performance and free software licenses.

Moreover, the Linux announcement is a strong signal that Microsoft has abandoned its longstanding ambition to make Windows the platform for everything related to computing. Microsoft CEO Satya Nadella had already, in many ways, signaled an attitude towards Linux that is very different from that of his predecessor Steve Ballmer. So how does SQL Server on Linux fit into the picture?

Let's split up the competitive position of SQL Server into three issues:

1. Its current marketing campaign to attract customers from Oracle.

2. The viability of SQL Server as a standalone piece of software on Linux.

3. The role of Linux, including a SQL Server port, in Microsoft's emerging cloud landscape.

As for the first item, it's easy to dismiss Microsoft's marketing campaign to try to get Oracle customers to migrate to SQL Server. Such campaigns have been common during the database wars over the last three decades and they typically have had very little to show for their efforts. It's hard enough just to upgrade a production instance of an Oracle database to a new version of Oracle and most customers wouldn't even contemplate migrating it to some other database platform. Databases are extremely sticky, and these marketing campaigns to target a competitor's database for migration have turned out to be little more than PR stunts that have

had very limited material success. Microsoft's latest entry into this game will likely be no different.

So it's really items 2 and 3 that are interesting and require a fair amount of discussion.

## The problems with SQL Server as a Linux product

Before getting into the change in Microsoft's global vision of the future roles of Windows and Linux, it may make sense to consider the merits of a Linux port for SQL Server as an isolated proposition. Would it make sense for Microsoft? Microsoft has ported some of its software to non-Windows platforms for a long time, e.g., Office for Mac, so it wouldn't be completely without precedent. On the other hand, Microsoft spent a lot of effort trying to kill Linux, but with that effort apparently over, why not try to generate some additional revenue on that platform? After all, a lot of the world's enterprise databases run Oracle or DB2 on Linux, so there would seem to be a market opportunity here.

One obvious problem with a Linux port would be the lack of credibility and skill pool. The hardcore Linux crowd typically doesn't hold Microsoft in all that high regard and the Windows crowd that knows SQL Server is rarely deeply skilled in Linux, so where are all the people that are going to be needed to set up and administer SQL Server databases on Linux?

A perhaps even bigger challenge is the value of SQL Server without being part of the Windows ecosystem. Whatever you may think about the quality of SQL Server purely from the standpoint of a relational database server, it's undeniable that a significant component of its value proposition in a Windows environment comes from its integration with other components in the same ecosystem.

The Windows ecosystem, by virtue of the integration of components, adds value that is more than the sum of its parts, whether they be the Windows OS itself, .NET, Active Directory, SQL Server, Excel, Analysis Services, the many other Microsoft components and numerous third-party tools and applications that run on Windows. And let's not forget all the professional services and the deep pool of talent that is readily available within the world of Windows. It's far from clear that extracting merely the "core relational database capabilities"
http://www.zdnet.com/article/microsoft-is-porting-sql-server-to-linux/
of SQL Server to be run in a completely different environment would make it even remotely as useful as within the Windows ecosystem.

But perhaps even more interesting are the technological challenges that come with the port itself.

Oracle has long been good at porting its database software to different platforms because since the very early days, the ability to run Oracle on any hardware or OS

has been used as a major selling point. The same has not been true of SQL Server and that will likely impose some challenges when it come to creating a viable Linux port.

The obvious approach to creating portable software is to distinguish generic code from code that has to be specific to the underlying OS or hardware. In the case of a relational database system, the generic code will likely include functionality like parsing SQL statements and evaluating a search space of potential join orders for the purpose of query optimization. But closer to the hardware, the way in which the database reads and writes data to disk or manages processes and threads will be more likely to include OS specific components. So a natural way to address a portable architecture is to separate the generic code from the OS dependent code by an API abstraction where there is a set of well-defined functionality that the generic code relies on but that is implemented separately on different platforms by porting organizations that have the required platform expertise.

Maintaining the right demarcation line between generic and OS specific code and the right API requires a certain amount of discipline. Oracle probably does as good a job of it as anyone could do and, historically, portability has been part of the reason for its success in the marketplace. However, even for the Oracle database, it's unlikely that all ports are going to work equally well.

One of the basic issues is that any piece of software is likely to run better on the platform on which it was developed than on any later port. The original developers presumably were very familiar with the platform and all potential platform-related issues and would make sure that the code would run really well on it. As the developers test their own code as they develop it, they would be much more likely to catch any issue, whether a pure bug or a performance issue, at an early stage. The developers who port the code to another platform, may well be different people who may have platform specific skills but didn't write the original code for the base platform. So they are supposed to take code they didn't write themselves and make it run well on a platform it wasn't developed on – not a trivial task.

But even in the best-case scenarios with extremely competent people taking care of the OS specific functionality, it's very likely that there will be differences between ports just because different platforms behave somewhat differently and that could have an impact on performance, if nothing else.

But let's not leave out functionality when it comes to the challenge of porting software. The documentation for the z/OS port of Oracle to IBM mainframes makes it clear that users should be aware of fundamental functionality issues that are specific to this Oracle port, like ASCII vs. EBCDIC encoding for character columns.

Yet another stumbling block for a good port of SQL Server to Linux may be a historic lack of discipline in terms of keeping Windows-specific code from making its way into what should be generic code for a portable database server. During the

Gates/Ballmer-era, the philosophy was that Windows would be the one and only platform for everything. It would seem unlikely that there was a strict discipline in terms of maintaining an abstraction of generic vs. port-specific code. If you assume that SQL Server will never run on anything besides Windows, there would be no reason not to cram in as many Windows-specific optimizations and as much Windows specific functionality as possible anywhere in the code.

So it seems likely that the SQL Server code, for the last two decades or so, has been developed without a whole lot of portability discipline and that the lack of such discipline will make the porting work harder. It will be interesting to see how much these issues impact the port and what functionality actually gets ported.

## The role of Linux in Microsoft's future

Here we have what may be a significant part of the explanation for the Linux port announcement – Microsoft's general push to embrace Linux in conjunction with its push to the cloud. There are two parts to this story, Microsoft's sense of the importance of the cloud and its sense that it needs Linux to compete.

1. Microsoft has realized that cloud platforms will be the new centers of gravity going forward. Much like Windows was an important ecosystem for computation in the past decades, the public cloud platform may play a similar roll in the future, complete with a vast selection of services, applications, and tools, skill pools and readily available professional services, and lots of third-party participation. As the cloud slowly renders the old world order obsolete, including the Windows franchise in its current form, Microsoft wants Azure to be a major player in the new world order.

2. Microsoft has realized that being Windows-only would relegate it to becoming a niche player in the cloud, albeit probably a very significant one. Microsoft failed in its efforts to stop Linux from becoming a major player in enterprise computing and would find itself at a major competitive disadvantage to AWS if it became regarded solely as a platform for diehard Microsoft shops. AWS, of course, has no intrinsic problems supporting any kind of OS a customer wants to run on. So rather than clinging to its dream of Windows as the dominant enterprise computing platform, Microsoft is going all in on Azure as a major player in the cloud, and that would be hard to do without a Linux component. Obviously, there is not a whole lot of money to be made by Microsoft by supporting Linux when viewed in isolation from its overall platform strategy – how lucrative would it be for Microsoft to support a bunch of pony-tailed, open-source enthusiasts running free software on Azure compared to traditional Microsoft shops running on the proprietary Windows stack? So Microsoft's Linux strategy is clearly based on long-term strategic concerns rather than short-term fat margins. Whether Microsoft's Linux effort can gain any significant credibility remains to be seen.

## Conclusion

Microsoft's overall cloud-related Linux push probably plays a role in its SQL Server strategy. How successful that Linux strategy will be, whether in the cloud or for SQL Server, remains a big question mark. It faces significant credibility and mindshare problems in both areas as well as numerous technical challenges for the SQL Server port. On the other hand, if you believe that the old Windows franchise is going to be marginalized by the cloud (and mobile) anyway, what has Microsoft got to lose by trying to play the Linux card? Probably not much. And even if the Linux strategy doesn't become a huge success, Microsoft's Windows niche in the cloud will likely still be a pretty nice consolation prize.