Chapter 1

The Evolution of Microsoft SQL Server: 1989 to 2000

In 1985, Microsoft and IBM announced "a long-term joint development agreement for development of operating systems and other systems software products." This announcement was the beginning of OS/2, a successor to the Microsoft MS-DOS operating system. OS/2 would be a more complete and robust operating system. It would exploit the powerful new personal computers based on the Intel 80286 processor. And it would allow multitasking applications, each with its own address space and each running in the safe ring 3 of the Intel four-ring protection scheme of the 80286. Machines sold in 1986 would be vastly more powerful than the original IBM PC (an Intel 8088-based machine) of just a couple years earlier, and OS/2 would signal a new age in harnessing this power. That was the plan.

OS/2 was formally announced in April 1987, with shipment promised by the end of the year. (OS/2 version 1.0 was released to manufacturing on December 16, 1987.) But shortly after the joint declaration, IBM announced a special higher-end version of OS/2 called OS/2 Extended Edition. This more powerful version would include the base OS/2 operating system plus an SQL RDBMS called OS/2 Database Manager. OS/2 Database Manager would be useful for small applications and would be partially compatible (although not initially interoperable) with DB/2, IBM's flagship MVS mainframe database, and with the lesser-used SQL/DS, which ran on slightly smaller mainframe computers using the VM or VSE operating systems. OS/2 Database Manager would also include Systems Network Architecture (SNA) communications services, called OS/2 Communications Manager. As part of its sweeping Systems Application Architecture (SAA), IBM promised to make the products work well together in the future. (OS/2 Database Manager later evolved into today's DB2/2.)

But if IBM could offer a more complete OS/2 solution, who would buy Microsoft OS/2? Clearly, Microsoft needed to come up with an answer to this question.

SQL Server: The Early Years

In 1986, Microsoft was a mere $197-million-per-year business with 1153 employees. (Ten years later, Microsoft had revenues of nearly $6 billion and almost 18,000 employees.) Microsoft's products were entirely desktop focused, and the main bread-and-butter product was MS-DOS. Client/server computing was not yet in the vernacular of Microsoft or the computer industry. Data management on PCs was in its infancy. Most people who kept data on their PCs used the wildly popular Lotus 1-2-3 spreadsheet application to keep lists (although many were discovering the limitations of doing so). Ashton-Tate's dBASE products (dBASE II and the recently released dBASE III) had also become popular. Although a few other products existed, such as MicroRim's Rbase and a relatively new product from Ansa Software called Paradox, Ashton-Tate was clearly king of the PC data products. In 1986, Microsoft had no database management products. (Beginning in 1992, Microsoft would go on to achieve tremendous success in the desktop database market with Microsoft Access and Microsoft FoxPro.)

IBM's Database Manager wasn't in the same category as products such as dBASE, Paradox, and Rbase. Database Manager was built to be a full-featured database (with atomic transactions and a full SQL query processor), more similar to traditional minicomputer-oriented or mainframe-oriented systems such as IBM's DB/2, or Oracle, or Informix. Microsoft needed a database management system (DBMS) product of the same caliber, and it needed one soon.

Microsoft turned to Sybase, Inc., an upstart in the DBMS market. Sybase hadn't yet shipped the first commercial version of its DataServer product (which it would do in May 1987 for Sun workstations running UNIX). Although certainly not a mainstream product, the prerelease version of DataServer had earned a good reputation for delivering innovative new capabilities, such as stored procedures and triggers, and because it had been designed for a new paradigm in computing: client/server.

As is true in all good business exchanges, the deal between the two companies was a win-win situation. Microsoft would get exclusive rights to the DataServer product for OS/2 and all other Microsoft-developed operating systems. Besides getting royalties from Microsoft, Sybase would get credibility from Microsoft's endorsement of its technology. Even more important, Sybase would gain a beachhead among the anticipated huge number of personal computers that would be running the new OS/2 operating system.

Because the transaction-processing throughput of these OS/2 systems wasn't expected to be high, Sybase could use the systems to seed the market for future sales on the more powerful UNIX system. Microsoft would market the product in higher volumes than Sybase could; it simply wasn't economically feasible for Sybase's direct sales force to deliver what would essentially be the first shrink-wrapped release of a full-fledged, blood-and-guts database to PC customers. Higher volumes would help Sybase win more business on UNIX and VMS platforms. On March 27, 1987, Microsoft president Jon Shirley and Sybase cofounder and president Mark Hoffman signed the deal.

In the PC database world, Ashton-Tate's dBASE still had the reputation and the lion's share of the market, even if dBASE and Sybase DataServer offered extremely different capabilities. To gain acceptance, this new, higher-capability DBMS from Microsoft (licensed from Sybase) would need to appeal to the large dBASE community. The most direct way to do that, of course, was to get Ashton-Tate to endorse the product—so Microsoft worked out a deal with Ashton-Tate to do just that.

In 1988, a new product was announced with the somewhat clumsy name *Ashton-Tate/Microsoft SQL Server*. Although not part of the product's title, Sybase was prominent in the product's accompanying information. This new product would be a port of Sybase DataServer to OS/2, marketed by both Ashton-Tate and Microsoft. Ashton-Tate had pledged that its much-anticipated dBASE IV would also be available in a server edition that would use the dBASE IV development tools and language as a client to develop applications (for example, order-entry forms) that would store the data in the new SQL Server product. This new client/server capability promised to give dBASE new levels of power to support more than the handful of concurrent users that could be supported by its existing file-sharing architecture.

Ashton-Tate, Microsoft, and Sybase worked together to debut SQL Server on OS/2. (This was the first use of the name *SQL Server*. Sybase later renamed its DataServer product for UNIX and VMS *Sybase SQL Server*. Today, Sybase's database server is known as *Sybase Adaptive Server*.)

The first beta version of Ashton-Tate/Microsoft SQL Server shipped in the fall of 1988. Microsoft made this prerelease version available at nominal cost to developers who wanted to get a head start on learning, developing for, or evaluating this new product. It shipped in a bundle known as the NDK (network development kit) that included all the necessary software components (provided that you were developing in C) for getting a head start on building networked client/server applications. It included prerelease versions of SQL Server, Microsoft LAN Manager, and OS/2 1.0.

Ron's Story

In 1988, Ron Soukup was working for Covia, the United Airlines subsidiary that provided the Apollo Reservation System and related systems for airport and travel agency use. He had spent the previous five years working with the new breed of relational database products that had appeared on the minicomputer and mainframe computer scene. He had worked with IBM's DB/2 running on MVS; IBM's SQL/DS running on VM; and Oracle, Informix, and Unify running on UNIX. Ron viewed PC databases of the day essentially as toys that were good for storing recipes and addresses but not much else. He used a PC for word processing, but that was about it. At work, they were beginning to use more and more LAN-based and PC-based systems, and Ron had begun doing some OS/2 programming. When he heard of the NDK, with this new SQL Server product that had been mentioned in the trade press, he ordered it immediately.

The NDK was very much a beta-level product. It didn't have a lot of fit and finish, and it crashed a couple of times a day. But from practically the first day, Ron knew that this product was something special. He was amazed that he was using a true DBMS—on a PC!—with such advanced features as transaction logging and automatic recovery. Even the performance seemed remarkably good. Having used mostly minicomputer and mainframe computer systems, he was most struck by the difference in PC response time. With the bigger systems, even a simple command resulted in an inevitable delay of at least a couple of seconds between pressing the Enter key and receiving a reply. PCs seemed almost instantaneous. Ron knew PCs were fast for local tasks such as word processing, but this was different. In this case, at one PC he entered an SQL command that was sent over the network to a server machine running this new SQL Server product. The response time was a subsecond. He had never seen such responsiveness.

Ron's initial kick-the-tires trial was encouraging, and he received approval to test the product more thoroughly. He wanted to get a feel for the types of applications and workloads for which this interesting new product might be used. For this, Ron wanted more substantial hardware than the desktop machine he originally tested on (a 10MHz 286 computer with 6 MB of memory and a 50-MB hard drive). Although SQL Server ran reasonably well on his desktop machine, he wanted to try it on one of the powerful new machines that used the Intel 80386 processor. Ron was able to procure a monster machine for the day—a 20-MHz 386 system with 10 MB of memory and two 100-MB disk drives—and was the envy of his division!

In 1987, Ron and a colleague at Covia had developed some multiuser database benchmark tests in C to help them choose a UNIX minicomputer system for a new application. So Ron dusted off these tests and converted the embedded C to the call-level interface provided in SQL Server (DB-Library) and ported these benchmarks to the PC. Ron hoped that SQL Server would handle several simultaneous users, although he didn't even consider that it could come close to handling the 15 to 20 simulated users they had tried in the earlier minicomputer tests. After many false starts and the typical problems that occur while running an early beta of a version 1.0 product, he persevered and got the test suite—2000 lines of custom C code—running on a PC against the beta version of SQL Server.

The results were amazing. This beta version of SQL Server, running on a PC that cost less than $10,000, performed as well and in many cases better than the minicomputer systems that they had tested a few months earlier. Those systems cost probably 10 times as much as the PC that Ron was using, and they needed to be managed by a professional UNIX system administrator. Ron knew the industry was in for a big change.

In May 1989, Ashton-Tate/Microsoft SQL Server version 1.0 shipped. Press reviews were good, but sales lagged. OS/2 sales were far below what had been expected. Most users hadn't moved from MS-DOS to OS/2, as had been anticipated. And about the only tool available to create SQL Server applications was C. The promised dBASE IV Server Edition from Ashton-Tate was delayed, and although several independent software vendors (ISVs) had promised front-end development tools for SQL Server, these hadn't materialized yet.

During the preceding six months, Ron had come to really know, respect, and admire SQL Server, and he felt the same about the people at Microsoft with whom he had worked during this period. So in late 1989, Ron accepted a position at Microsoft in the SQL Server group in Redmond, Washington. A few months later, he was running the small but talented and dedicated SQL Server development team.

Kalen's Story

Kalen Delaney saw the first version of Microsoft SQL Server at about the same time that Ron did. She started working for Sybase Corporation's technical support team in 1987. And in 1988, she started working with a team that was testing the new PC-based version of Sybase SQL Server, which a company in the Seattle area was developing for OS/2. Sybase would be providing phone support for customers purchasing this version of the product, and Kalen, along with the rest of the technical support team, wanted to be ready.

Up to that point, she had never actually worked with PC-based systems and had pegged them as being mere toys. Sybase SQL Server was an extremely powerful product, capable of handling hundreds of simultaneous users and hundreds of megabytes of data. Kalen supported customers using Sybase SQL Server databases to manage worldwide financial, inventory, and human resource systems. A serious product was required to handle these kinds of applications, and Kalen thought PCs just didn't fit the bill. However, when she saw a PC sitting on a coworker's desk actually running SQL Server, Kalen's attitude changed in a hurry. Over the next few years, she worked with hundreds of customers in her product support position and later in her position as Senior Instructor, using Microsoft SQL Server to manage real businesses with more than toy amounts of data, from a PC.

In 1991, Kalen and her family left the San Francisco Bay Area to live in the Pacific Northwest, but she continued to work for Sybase as a trainer and courseware developer. A year later, Sybase dissolved her remote position, and she decided to start her own business. A big decision loomed: should she focus primarily on the Sybase version or the Microsoft version of SQL Server? Because Microsoft SQL Server would run on a PC that she could purchase at her local Sears store with low monthly payments, and because she could buy SQL Server software and the required operating system in one package, the decision was simple. Microsoft had made it quite easy for the sole proprietor to acquire and develop software and hone his or her skills on a powerful, real-world, relational database system.

Microsoft SQL Server Ships

By 1990, the co-marketing and distribution arrangement with Ashton-Tate, which was intended to tie SQL Server to the large dBASE community, simply wasn't working. Even the desktop version of dBASE IV was quite late, and it had a reputation for being buggy when it shipped in 1989. The Server Edition, which would ostensibly make it simple to develop higher-performance SQL Server applications using dBASE, was nowhere to be seen.

As many others have painfully realized, developing a single-user, record-oriented application is quite different from developing applications for multiple users for which issues of concurrency, consistency, and network latency must be considered. Initial attempts at marrying the dBASE tools with SQL Server had dBASE treating SQL Server as though it were an indexed sequential access method (ISAM). A command to request a specific row was issued for each row needed. Although this procedural model was what dBASE users were accustomed to, it wasn't an efficient way to use SQL Server, with which users could gain more power with less overhead by issuing SQL statements to work with sets of information. But at the time, SQL Server lacked the capabilities to make it easy to develop applications that would work in ways dBASE users were accustomed to (such as browsing through data forward and backward, jumping from record to record, and updating records at any time). Scrollable cursors didn't exist yet.

NOTE

The effort to get dBASE IV Server Edition working well provided many ideas for how scrollable cursors in a networked client/server environment should behave. In many ways, it was the prime motivation for including this feature in SQL Server version 6.0 in 1995, six years later.

Only two years earlier, Ashton-Tate had been king of the PC database market. Now it was beginning to fight for its survival and needed to refocus on its core dBASE desktop product. Microsoft would launch OS/2 LAN Manager under the Microsoft name (as opposed to the initial attempts to create only OEM versions), and it needed SQL Server to help provide a foundation for the development of client/server tools that would run on Microsoft LAN Manager and Microsoft OS/2. So Microsoft and Ashton-Tate terminated their co-marketing and distribution arrangements. The product would be repackaged and reintroduced as Microsoft SQL Server.

Microsoft SQL Server version 1.1 shipped in the summer of 1990 as an upgrade to the Ashton-Tate/Microsoft SQL Server version 1.0 that had shipped in 1989. For the first time, SQL Server was a Microsoft-supported, shrink-wrapped product, and it was sold through the newly formed Microsoft Network Specialist channel, whose main charter was to push Microsoft LAN Manager.

NOTE

When version 1.1 shipped, Microsoft didn't see SQL Server as a lucrative product in its own right. SQL Server would be one of the reasons to buy LAN Manager—that's all.

SQL Server 1.1 had the same features as version 1.0, although it included many bug fixes—the type of maintenance that is understandably necessary for a version 1.0 product of this complexity. But SQL Server 1.1 also supported a significant new client platform, Microsoft Windows 3.0. Windows 3.0 had shipped in May 1990, a watershed event in the computer industry. SQL Server 1.1 provided an interface that enabled Windows 3.0-based applications to be efficiently developed for it. This early and full support for Windows 3.0-based applications proved to be vital to the success of Microsoft SQL Server. The success of the Windows platform would also mean fundamental changes for Microsoft and SQL Server, although these changes weren't yet clear in the summer of 1990.

With the advent of Windows 3.0 and SQL Server 1.1, many new Windows-based applications showed up and many were, as promised, beginning to support Microsoft SQL Server. By early 1991, dozens of third-party software products used SQL Server. SQL Server was one of the few database products that provided a Windows 3.0 dynamic link library (DLL) interface practically as soon as Windows 3.0 shipped, which was now paying dividends in the marketplace. Quietly but unmistakably, Microsoft SQL Server was the leading database system used by Windows applications. Overall sales were still modest, but until tools beyond C existed to build solutions, sales couldn't be expected to be impressive. It was the classic chicken-and-egg situation.

Development Roles Evolve

Microsoft's development role for SQL Server 1.0 was quite limited. As a small porting team at Sybase moved its DataServer engine to OS/2 and moved the DB-Library client interfaces to MS-DOS and OS/2, Microsoft provided testing and project management. Microsoft also developed some add-on tools to help make SQL Server 1.0 easy to install and administer.

Although a number of sites were running OS/2 as an application server with SQL Server or as a file server with LAN Manager, few were using OS/2 for their desktop platforms. Before Windows 3.0, most desktops remained MS-DOS based, with a well-known limit of 640 KB of addressable real memory. After loading MS-DOS, a network redirector, network card device drivers, and the DB-Library static link libraries that shipped with SQL Server version 1.0, developers trying to write a SQL Server application were lucky to get 50 KB for their own use.

For SQL Server 1.1, rather than shipping the DB-Library interface that Sybase had ported from UNIX to MS-DOS, Microsoft wrote its own, from scratch. Instead of 50 KB, developers could get up to 250 KB to write their applications. Although small by today's standards, 250 KB was a huge improvement.

> **NOTE**
>
> The same source code used for DB-Library for MS-DOS also produced the Windows and OS/2 DB-Library interfaces. But the MS-DOS RAM cram is what motivated Microsoft to write a new implementation from scratch. The widespread adoption of Windows 3.0 quickly eliminated the MS-DOS memory issue—but this issue was a real problem in 1989 and 1990.

With SQL Server 1.1, Microsoft provided client software and utilities, programming libraries, and administration tools. But the core SQL Server engine was still produced entirely by Sybase; Microsoft didn't even have access to the source code. Any requests for changes, even for bug fixes, had to be made to Sybase.

Microsoft was building a solid support team for SQL Server. It hired some talented and dedicated engineers with database backgrounds. But with no access to source code, the team found it impossible to provide the kind of mission-critical responsiveness that was necessary for customer support. And, again, fixing bugs was problematic because Microsoft depended entirely on Sybase, which had become successful in its own right and was grappling with its explosive growth. Inevitably, some significant differences arose in prioritizing which issues to address, especially when some issues were specific to the Microsoft-labeled product and not to Sybase's product line. Bugs that Microsoft deemed of highest priority sometimes languished because Sybase's priorities were understandably directed elsewhere. The situation was unacceptable.

It was a great day in the SQL Server group at Microsoft when, in early 1991, Microsoft's agreement with Sybase was amended to give Microsoft read-only access to the source code for the purpose of customer support. Although Microsoft's SQL Server group still couldn't fix bugs, they at least could read the source code to better understand what might be happening when something went wrong. And they could also read the code to understand how something was expected to work. As anyone with software development experience knows, even the best specification is ambiguous at times. There's simply no substitute for the source code as the definitive explanation for how something works.

As a small group of developers at Microsoft became adept with the SQL Server source code and internal workings, Microsoft began to do virtual bug fixes. Although they still weren't permitted to alter the source code, they could identify line-by-line the specific modules that needed to be changed to fix a bug.

Obviously, when they handed the fix directly to Sybase, high-priority bugs identified by Microsoft were resolved much more quickly.

After a few months of working in this way, the extra step was eliminated. By mid-1991, Microsoft could finally fix bugs directly. Because Sybase still controlled the baseline for the source code, all fixes were provided to Sybase for review and inclusion in the code. The developers at Microsoft had to make special efforts to keep the source code highly secured, and the logistics of keeping the source code in sync with Sybase was sometimes a hassle, but all in all, this was *heaven* compared to a year earlier. Microsoft's team of developers was becoming expert in the SQL Server code, and they could now be much more responsive to their customers and responsible for the quality of the product.

OS/2 and Friendly Fire

In 1991, Microsoft released SQL Server 1.11, a maintenance release. SQL Server was slowly but steadily gaining acceptance and momentum—and a long list of ISV supporters. Client/server computing wasn't widely deployed yet, but new converts appeared every day. Customer satisfaction and loyalty were high, and press reviews of the product were all favorable. Sales were generally disappointing, but this was hardly a surprise because OS/2 had continued to be a major disappointment. Windows 3.0, however, was a runaway hit. Rather than move their desktop platforms from MS-DOS to OS/2, huge numbers of PC users moved to Windows 3.0 instead. OS/2 hadn't become a widespread operating system as anticipated, and it was now abundantly clear that it never would be.

SQL Server's Limitations and the Marketplace

~~Microsoft SQL Server 1.11 clearly had a scalability limit. It was a 16-bit product because OS/2~~ could provide only a 16-bit address space for applications, and its throughput was hampered by OS/2's lack of some high-performance capabilities, such as asynchronous I/O. Even though an astonishing amount of work could be performed successfully with SQL Server on OS/2, there would come a point at which it would simply run out of gas. No hard limit was established, but in general, SQL Server for OS/2 was used for workgroups of 50 or fewer users. For larger groups, customers could buy a version of Sybase SQL Server for higher-performance UNIX-based or VMS-based systems.

This was an important selling point for both Microsoft and Sybase. Customers considering the Microsoft product wanted to be sure they wouldn't outgrow it. The large number of ISV tools developed for Microsoft SQL Server worked largely unchanged with Sybase SQL Server, and applications that outgrew OS/2 could be moved quite easily to a bigger, more powerful, more expensive UNIX system. This relationship still made sense for both Microsoft and Sybase.

The need for compatibility and interoperability made it especially important for Microsoft SQL Server to be based on the version 4.2 source code as soon as possible. Furthermore, a major features version hadn't been released since version 1.0 in 1989. In the rapidly moving PC marketplace, the product was in danger of becoming stale. Customers had begun to do serious work with Microsoft SQL Server, and new features were in great demand. Microsoft's version 4.2 would add a long list of significant new features, including server-to-server stored procedures, UNION, online tape backup, and greatly improved international support that would make SQL Server more viable outside the United States.

At the same time, Microsoft was working on a new SQL Server version that would sync up with the newest Sybase product on UNIX, version 4.2. When Microsoft SQL Server 1.0 shipped, Sybase's product was designated version 3.0. They had added some new features deemed necessary for the PC marketplace, such as *text* and *image* datatypes and browse mode. Sybase subsequently shipped version 4.0 for most platforms and version 4.2 on a more limited basis.

Meanwhile, in May 1991, Microsoft and IBM announced an end to their joint development of OS/2. Clearly, most customers were voting with their dollars for Windows, not OS/2. Microsoft decided to concentrate on future versions of Windows and applications for Windows. The announcement, although not a surprise, rocked the industry nonetheless. Microsoft was well underway in the development of a new microkernel-based operating system that was internally code-named *NT* (for "new technology"). This new system was originally envisioned as a future release of OS/2 and was sometimes referred to as *OS/2 3.0*. After the termination of joint OS/2 development, the NT project was altered to include the Windows user interface and the Win32 application programming interface (API), and it became known henceforth as Microsoft Windows NT.

The first version of Windows NT wasn't expected for two years. Microsoft SQL Server would eventually be moved to Windows NT—that was a no-brainer. But in the meantime, Microsoft had to continue developing SQL Server on OS/2, even though OS/2 was now a competitive product for Microsoft; there was no alternative. For the next couple of years, the SQL Server group got used to getting hit by friendly fire as Microsoft competed vigorously against OS/2.

SQL Server 4.2

Microsoft had been developing SQL Server 4.2 for the forthcoming OS/2 2.0, the first 32-bit version of OS/2. Since SQL Server 4.2 was also planned to be 32-bit, porting the product from its UNIX lineage would be easier because memory segmentation issues wouldn't be a concern. In theory, the 32-bit SQL Server would also perform faster. Many articles comparing 32-bit and 16-bit performance issues appeared in the press, and everyone assumed that the 32-bit environment would bring awesome performance gains (although few articles explained correctly why this might or might not be true).

The principal performance gain expected would be due to memory addressing. To address memory in the 16-bit segmented address space of OS/2 1.*x*, basically two instructions were required: one to load the correct segment and one to load the memory address within that segment. With 32-bit addressing, the instruction to load the segment was unnecessary and memory could be addressed with one instruction, not two. Because addressing memory is so common, some quick calculations showed that the 32-bit version might yield an overall performance increase of perhaps 20 percent or more, if all other operations were of equal speed.

The 32-Bit Platform: Bigger Isn't Always Better

Many people mistakenly believed that SQL Server needed to be running on a fully 32-bit platform to address more than 16 MB of memory. Under OS/2 1.*x*, an application could access a maximum of 16 MB of real memory; it could access more than 16 MB of *virtual* memory, but paging would result. In OS/2 2.0, an application could access more than 16 MB of real memory and avoid paging. This would allow SQL Server to have a larger cache, and getting data from memory rather than from disk always results in a huge performance gain. However, to the application, all memory in OS/2 was virtual memory—in both versions. So even the 16-bit version of SQL Server would be able to take advantage of the ability of OS/2 2.0 to access larger real-memory spaces. A 32-bit version was unnecessary for this enhancement.

Unfortunately, the early beta versions of OS/2 2.0 were significantly slower than OS/2 1.*x*, more than offsetting the efficiency in addressing memory. So rather than a performance gain, users saw a significant loss of performance in running Microsoft SQL Server 1.1 and preliminary builds of the 32-bit SQL Server 4.2 on OS/2 2.0.

OS/2 2.0 Release on Hold

Suddenly, the plans for the release of OS/2 2.0 by the end of 1991 were suspect. In fact, it was unclear whether IBM could deliver version 2.0 at all. At any rate, it appeared doubtful that OS/2 would ship any sooner than the end of 1992. The decision became clear—Microsoft would move SQL Server back to a 16-bit implementation and target it for OS/2 1.3.

Reworking back to 16-bit would cost the Microsoft developers about three months, but they had little choice. In the meantime, another problem appeared. IBM shipped OS/2 1.3, but this version worked only for its brand of PCs. Theoretically, other computer manufacturers could license OS/2 from Microsoft and ship it as part of an OEM agreement. However, the demand for OS/2 had become so small that most OEMs didn't ship it; as a result, buying OS/2 for other PCs was difficult for customers. For the first time, despite the seeming incongruity, Microsoft produced a shrink-wrapped version of OS/2 version 1.3, code-named *Tiger*. Tiger shipped in the box with Microsoft SQL Server and Microsoft LAN Manager, lessening the problem that the product was essentially targeted to a dead operating system.

Version 4.2 Released

Microsoft SQL Server version 4.2 entered beta testing in the fall of 1991, and in January 1992, Microsoft CEO Bill Gates (with Sybase's Bob Epstein sharing the stage) formally announced the product at a Microsoft SQL Server developers' conference in San Francisco. Version 4.2 truly had been a joint development between Microsoft and Sybase. The database engine was ported from the UNIX version 4.2 source code, with both Microsoft and Sybase engineers working on the port and fixing bugs. In addition, Microsoft produced the client interface libraries for MS-DOS, Windows, and OS/2, and for the first time it included a Windows GUI tool to simplify administration. Source code for the database engine was merged back at Sybase headquarters, with files exchanged via modem and magnetic tape.

Microsoft SQL Server version 4.2 shipped in March 1992. The reviews were good and customer feedback was positive. As it turned out, the source code for the database engine in this version was the last code that Microsoft received from Sybase (except for a few bug fixes exchanged from time to time).

After the product shipped, the Big Question for 1992 was, "When will a 32-bit version of Microsoft SQL Server be available?" The issue of 32-bitness became an emotional one. Many people who were the most strident in their demands for it were unclear or confused about what the benefits would be. They assumed that it would automatically have a smaller footprint, run faster, address more memory, and generally be a much higher-performing platform. But the internal work with a 32-bit version for OS/2 2.0 had shown that this wouldn't necessarily be the case.

SQL Server for Windows NT

Contrary to what many people might assume, Microsoft senior management never pressured the SQL Server development team to *not* develop a full 32-bit version for OS/2. One of the joys of working for Microsoft is that senior management empowers the people in charge of projects to make the decisions, and this decision was left with Ron Soukup.

In early 1992, however, the development team faced some uncertainty and *external* pressures. On one hand, the entire customer base was, by definition, using OS/2. Those customers made it quite clear that they wanted—indeed, expected—a 32-bit version of SQL Server for OS/2 2.0 as soon as IBM shipped OS/2 2.0, and they intended to remain on OS/2 in the foreseeable future. But when OS/2 2.0 would be available was unclear. IBM claimed that it would ship by the fall of 1992. Steve Ballmer, Microsoft senior vice president, made a well-known pledge that he'd eat a floppy disk if IBM shipped the product in 1992.

The SQL Server team was now under pressure to have a version of SQL Server running on Windows NT as soon as possible, with prerelease beta versions available when Windows NT was prereleased. The SQL Server team members knew that Windows NT was the future. It would be the high-end operating system solution from Microsoft, and from a developer's perspective, Windows NT would offer a number of technical advantages over OS/2, including asynchronous I/O, symmetric multiprocessing, and portability to reduced instruction set computing (RISC) architectures. They were champing at the bit to get started.

Although Microsoft had decided in 1991 to fall back to a 16-bit version of SQL Server, Microsoft's developers had continued to work on the 32-bit version. By March 1992, just after shipping version 4.2, they saw that both the 16-bit and 32-bit versions ran slower on the beta versions of OS/2 2.0 than the 16-bit version ran on Tiger (OS/2 1.3). Perhaps by the time OS/2 2.0 actually shipped, it might run faster. But then again, it might not—the beta updates of OS/2 2.0 didn't give any indication that it was getting faster. In fact, it seemed to be getting slower and more unstable.

For a product with the scope of SQL Server, there's no such thing as a small release. There are big releases and bigger releases. Because resources are finite, the developers knew that working on a product geared toward OS/2 2.0 would slow down the progress of Windows NT development and could push back its release. Adding more developers wasn't a good solution. (As many in the industry have come to learn, adding more people is often the cause of, and seldom the solution to, software development problems.) Furthermore, if Microsoft chose to develop simultaneously for both OS/2 2.0 and Windows NT, the developers would encounter another set of problems. They would have to add an abstraction layer to hide the differences in the operating systems, or they'd need substantial reengineering for both, or they'd simply take a lowest-common-denominator approach and not fully use the services or features of either system.

So Microsoft developers decided to stop work on the 32-bit version of SQL Server for OS/2 2.0. Instead, they moved full-speed ahead in developing SQL Server for Windows NT, an operating system with an installed base of zero. They didn't constrain the architecture to achieve portability back to OS/2 or to any other operating system. They vigorously consumed whatever interfaces and services Windows NT exposed. Windows NT was the only horse, and the development team rode as hard as they could. Except for bug fixing and maintenance, development ceased for SQL Server for OS/2.

Microsoft began to inform customers that future versions, or a 32-bit version, of SQL Server for OS/2 2.0 would depend on the volume of customer demand and that the primary focus was now on Windows NT. Most customers seemed to understand and accept this position, but for customers who had committed their businesses to OS/2, this was understandably not the message that they wanted to hear.

At this time, Sybase was working on a new version of its product, to be named System 10. As was the case when they were working on version 4.2, the developers needed Microsoft SQL Server to be compatible with and have the same version number as the Sybase release on UNIX. OS/2 vs. Windows NT was foremost at Microsoft, but at Sybase, the success of System 10 was foremost.

Although System 10 wasn't even in beta yet, a schedule mismatch existed between the goal of getting a version of Microsoft SQL Server onto Windows NT as soon as possible and getting a version of System 10 onto Windows NT, OS/2 2.0, or both as soon as possible. The development team decided to compromise and specialize. Microsoft would port SQL Server version 4.2 for OS/2 to Windows NT, beginning immediately. Sybase would bring Windows NT under its umbrella of core operating systems for System 10. Windows NT would be among the first operating system platforms on which System 10 would be available. In addition, Microsoft would turn the OS/2 product back over to Sybase so those customers who wanted to stay with OS/2 could do so. Although Microsoft hoped to migrate most of the installed customer base to Windows NT, they knew that this could never be 100 percent. They were honestly pleased to offer those customers a way to continue with their OS/2 plans, via Sybase. The SQL Server development team truly didn't want them to feel abandoned.

This compromise and specialization of development made a lot of sense for both companies. The development team at Microsoft would be working from the stable and mature version 4.2 source code that they had become experts in. Porting the product to a brand new operating system was difficult enough, even *without* having to worry about the infant System 10 code base. And Sybase could concentrate on the new System 10 code base without worrying about the inevitable problems of a prebeta operating system. Ultimately, System 10 and SQL Server for Windows NT would both ship, and the two companies would again move back to joint development. That was the plan, and in 1992 both sides expected this to be the case.

The SQL Server team at Microsoft started racing at breakneck speed to build the first version of SQL Server for Windows NT. Time to market was, of course, a prime consideration. Within Microsoft, the team had committed to shipping within 90 days of the release of Windows NT; within the development group, they aimed for 30 days. But time to market wasn't the only, or even chief, goal. They wanted to build the best database server for Windows NT that they could. Because Windows NT was now SQL Server's only platform, the developers didn't need to be concerned about portability issues, and they didn't need to create an abstraction layer that would make all operating systems look alike. All they had to worry about was doing the best job possible with Windows NT. Windows NT was designed to be a portable operating system, and it would be available for many different machine architectures. SQL Server's portability layer would be Windows NT itself.

The development team tied SQL Server into management facilities provided by Windows NT, such as raising events to a common location, installing SQL Server as a Windows NT service, and exporting performance statistics to the Windows NT Performance Monitor. Because Windows NT allows applications to dynamically load code (using DLLs), an open interface was provided to allow SQL Server to be extended by developers who wanted to write their own DLLs.

This first version of Microsoft SQL Server for Windows NT was far more than a port of the 4.2 product for OS/2. Microsoft essentially rewrote the *kernel* of SQL Server—the portion of the product that interacts with the operating system—directly to the Win32 API.

Another goal of SQL Server for Windows NT was to make it easy to migrate current installations on OS/2 to the new version and operating system. The developers wanted all applications that were written to SQL Server version 4.2 for OS/2 to work unmodified against SQL Server for Windows NT. Because Windows NT could be dual-booted with MS-DOS or OS/2, the team decided that SQL Server for Windows NT would directly read from and write to a database created for the OS/2 version. During an evaluation phase, for instance, a customer could work with the OS/2 version, reboot the same machine, work with the Windows NT version, and then go back to OS/2. Although it would be difficult to achieve, the developers wanted 100 percent compatibility.

The developers reworked SQL Server's internals and added many new management, networking, and extensibility features; however, they didn't add new external core database engine features that would compromise compatibility.

The Windows NT-Only Strategy

Many have questioned the Windows NT-only strategy. But in 1992, the UNIX DBMS market was already overcrowded, so Microsoft developers felt they wouldn't bring anything unique to that market. They recognized that SQL Server couldn't even be in the running when UNIX was a customer's only solution. Microsoft's strategy was based on doing the best possible job for Windows NT, being the best product on Windows NT, and helping to make Windows NT compelling to customers.

The decision to concentrate on only Windows NT has had far-reaching effects. To be portable to many operating systems, the Sybase code base had to take on or duplicate many operating system services. For example, because threads either didn't exist on many UNIX operating systems or the thread packages differed substantially, Sybase had essentially written its own thread package into SQL Server code. The Windows NT scheduling services, however, were all based on a thread as the schedulable unit. If multiprocessors were available to the system and an application had multiple runnable threads, the application automatically became a multiprocessor application. So the Microsoft SQL Server developers decided to use native Windows NT threads and not the Sybase threading engine.

The development team made similar choices for the use of asynchronous I/O, memory management, network protocol support, user authentication, and exception handling. (In Chapter 3, we'll look at the SQL Server architecture in depth and delve into more of these specifics. The point here is that the goals intrinsic to portability are in conflict with the goal to create the best possible implementation for a single operating system.)

For example, the developers ensured that there were no differences in the SQL dialect or capabilities of the Windows NT and OS/2 versions. The plan was for the future System 10 version to implement many new features. This release would be fundamentally a platform release. To emphasize compatibility with the OS/2-based 4.2 product and with the Sybase product line, Microsoft decided to call the first version of SQL Server for Windows NT *version 4.2*. The product was referred to as simply *Microsoft SQL Server for Windows NT* and often internally as *SQL NT*, a designation that the press and many customers also were beginning to use.

In July 1992, Microsoft hosted a Windows NT developers' conference and distributed a prebeta version of Windows NT to attendees. Even though SQL Server didn't exist yet at a beta level, Microsoft immediately made available via CompuServe the 32-bit programming libraries that developers needed for porting their applications from OS/2 or 16-bit Windows to Windows NT. Just as it had enjoyed success back in 1990 by providing the NDK to prospective Windows 3.0 developers, Microsoft sought the same success by providing all the necessary tools to would-be Windows NT developers.

Commitment to Customers

Incidentally, at approximately the same time they were delivering the NDK to developers, Microsoft also shipped a maintenance release of SQL Server for OS/2 (and they shipped another the following year). In porting to Windows NT, the development team found and fixed many bugs that were generic to all platforms. Even though they wouldn't create new SQL

Server versions for OS/2, they really meant it when they said that they didn't want to abandon the OS/2 customers.

By March 1993, Microsoft went a step further and made a public beta release; anyone (even competitors) could obtain this prerelease product, SQL Server Client/Server Development Kit (CSDK), for a nominal charge that essentially covered expenses. Microsoft set up a public support forum on CompuServe and didn't demand that participants sign a nondisclosure agreement. It shipped more than 3000 CSDKs. By May 1993, the volume on the support forum for the prerelease product exceeded that for the shipping OS/2 product. The feedback was highly positive: Microsoft had a winner. The dream of an eventual "client/server for the masses" was being realized.

In July 1993, Microsoft shipped Windows NT 3.1. Within 30 days, achieving their internal goal, the SQL Server development team released the first version of Microsoft SQL Server for Windows NT to manufacturing. Customer and press reaction was terrific. SQL Server for Windows NT was listed in many publications as among the top and most important new products of 1993.

In October 1992, Microsoft shipped the first beta version of SQL Server for Windows NT. The product was essentially feature-complete and provided a full Win32 version of all components. It shipped to more than 100 beta sites. For a database server, having 100 beta sites was unprecedented; the typical number of beta sites for such a product was approximately 10.

Success Brings Fundamental Change

SQL Server for Windows NT was a success by nearly all measures. The strategy of integrating the product tightly with Windows NT resulted in a product that was substantially easier to use than high-end database products had ever been. Sales were above the internal projections and increased as Windows NT began to win acceptance.

By early December 1993, much of the SQL Server customer base for the OS/2 operating system had already migrated to SQL Server for Windows NT. Surveys showed that most of those who hadn't yet upgraded to Windows NT planned to do so, despite the fact that Sybase had publicly announced its intention to develop System 10 for OS/2.

The upgrade from SQL Server for OS/2 to SQL Server for Windows NT was virtually painless. When applications were moved from one platform to the other, not only did they still work, but they worked better. SQL Server for Windows NT was much faster than SQL Server for OS/2, and most significant, it was scalable far beyond the limits imposed by OS/2. Within another nine months, Microsoft's SQL Server business had more than doubled, with nearly all sales coming on the Windows NT platform. Although they continued to offer the OS/2 product, it accounted for well below 10 percent of sales.

A Faster SQL Server

Focusing the product on Windows NT had made the product fast. Studies conducted internally at Microsoft, as well as those based on private customer benchmarks, all showed similar results: SQL Server for Windows NT (running on low-cost commodity hardware) was competitive in performance with database systems running on UNIX (much more costly hardware).

In September 1993, Compaq Computer Corporation published the first official, audited Transaction Processing Council (TPC) benchmarks. At that time, on the TPC-B benchmark, well over $1000/TPS (transactions per second) was common, and it was an impressive number that broke the $1000/TPS barrier. Running on a dual-Pentium, 66-MHz machine, SQL Server achieved 226 TPS at a cost of $440 per transaction, less than half the cost of the lowest benchmark ever published by any other company. The raw performance number of 226 TPS was equally astonishing. At that time, most of the TPC-B numbers on file for UNIX minicomputer systems were still below 100 TPS. Certainly a handful of numbers were higher, but all occurred at a price point of much more than $440/TPS. Just 18 months prior, the raw performance of 226 TPS was about as high as any mainframe or minicomputer system had ever achieved.

The implications were clear. SQL Server for Windows NT wasn't simply a low-end or workgroup system. Its performance was competitive with more costly UNIX systems, and the trend toward faster systems running Windows NT was unmistakable. The 66-MHz Pentium processors were the first generation of Pentiums from Intel. Much higher clock speeds were expected to emerge within a few months; hardware with additional processors was anticipated. Furthermore, Microsoft SQL Server for Windows NT would soon be available on RISC processors such as the DEC Alpha-AXP at 250 MHz and the MIPS R4400. The so-called *Moore's Law* (named after Gordon Moore, cofounder of Intel Corporation), which postulates that computing power doubles every 18 months, was clearly being proven true for the type of commodity hardware for which Windows NT and Microsoft SQL Server were designed. Microsoft took a serious look at what would be required to achieve 1000 TPS, definitely putting raw performance in the same league as even the largest systems of the day.

The End of Joint Development

Microsoft's success strained its relationship with Sybase. The competitive landscape of late 1993 was quite different from that of 1987, when Microsoft and Sybase had inked their deal. By 1993, Sybase was a successful software company, by most accounts second only to Oracle in the DBMS market. The credibility and visibility Microsoft brought to Sybase was far less important in 1993 than it had been to the upstart company of 1987. Similarly, Microsoft had grown a great deal since 1987. The growth wasn't just in revenues (although that was one of the great success stories of the industry) but also in the development of products for use with enterprise applications, such as Microsoft SQL Server, that Fortune 1000 companies could use as a platform on which to run their businesses.

The SQL Server development team had grown as well, from a handful of people in 1990 to more than 50 professionals (not including those in marketing, support, or field operations), with significant additional growth planned. The first-rate team of engineers knew database and transaction processing and the inner workings of SQL Server, and they were experts in developing for Windows NT. Microsoft now had the talent, size, motivation, and mandate, but it was still constrained in what it could do with the product: the 1987 agreement with Sybase had merely licensed to Microsoft the rights to the Sybase product. Because of this restricted agreement, Microsoft couldn't implement new features or changes without Sybase's approval. Contrary to what many people thought, Microsoft had no ownership stake in Sybase and could by no means simply call the shots.

Obviously, Sybase had different business needs and therefore different priorities than Microsoft. The development team at Microsoft might, for example, want to integrate SQL Server with messaging by using MAPI (Messaging API), but because this feature was specific to the Microsoft operating system, Sybase wouldn't be excited about it. As is always the case in development, many features *could* be implemented for every feature that is actually implemented: features specific to Windows NT didn't tend to interest Sybase as much as those that would benefit its UNIX products.

Sybase engineers had to confront the issue of portability to multiple operating systems. In fact, Microsoft's implementation of version 4.2 for Windows NT was already causing friction because Sybase was progressing with System 10 for Windows NT. Sybase was understandably implementing System 10 in a more portable manner than Microsoft had done. This was entirely rational for Sybase's objectives, but from Microsoft's perspective, it meant a looser bond with Windows NT. System 10 would not, and *could not,* perform as well on Windows NT as the product that had been designed and written exclusively for Windows NT.

Because of the economics involved, as well as the changing competitive landscape, the Microsoft/Sybase agreement of 1987 was no longer working. Microsoft SQL Server was now a viable competitive alternative to Sybase SQL Server running on UNIX, Novell NetWare, and VMS. Far from seeding the market for Sybase, Microsoft SQL Server was now taking sales away from Sybase. Instead of choosing a UNIX solution, customers could buy Microsoft SQL Server at a fraction of the cost of a UNIX solution, run it on less expensive PC hardware, and install and administer it more easily. Although Sybase earned royalties on sales of Microsoft SQL Server, this amounted to a small fraction of the revenue Sybase would have received if customers had bought Sybase products for UNIX in the first place. Microsoft and Sybase were now often vigorously competing for the same customers. Both companies recognized that a fundamental change in the relationship was needed.

On April 12, 1994, Microsoft and Sybase announced an end to joint development. Each company decided to separately develop its own SQL Server products. Microsoft was free to evolve and change Microsoft SQL Server. Sybase decided to bring its System 10 products (and subsequent versions) to Windows NT—the first time that the Sybase-labeled SQL Server would be available for a Microsoft operating system. (The original agreement gave Microsoft exclusive rights on Microsoft operating systems.) Both companies' products would be backward-compatible with existing SQL Server applications—however, the products would diverge in the future and would have different feature sets and design points. Sybase's product would be fully compatible with its UNIX versions. Microsoft's product would continue to be integrated with Windows NT as much possible. In short, the products would directly compete.

SQL Server Performance: No Secret Formulas

Although Microsoft SQL Server is designed for and optimized for Windows NT, it uses only publicly documented interfaces. From time to time, articles or speakers suggest that Microsoft SQL Server uses private, undocumented APIs in Windows NT to achieve its performance. But this assumption is false, without exception. SQL Server's performance results from using the available and published Windows NT services without any compromise. This is something that other products could also do if developers were willing to dedicate themselves to doing the best possible job for Windows NT without making compromises for portability to other operating systems.

NOTE

Although the Sybase product competed with Microsoft SQL Server, Microsoft encouraged and provided support for getting Sybase System 10 shipping on Windows NT as soon as possible because it was important to the acceptance and success of Windows NT. Such collaboration is typical; many such relationships are competitive on one level and cooperative on another.

The Charge to SQL95

As late as the beginning of 1994, the SQL Server development team had planned for the next version of the product to pick up the Sybase System 10 source code and new features. But the termination of joint development changed that plan. Sybase contributed no more code, and Microsoft released no System 10 product. Except for a couple of bug fixes, the last code drop received from Sybase was in early 1992 just as version 4.2 for OS/2 was shipping.

Time was at a premium. Besides promoting sales to new customers, Microsoft had to fight for its significant installed base. Sybase was to deliver System 10 for Windows NT later that year. That would be a potentially easy upgrade for Microsoft's installed customer base; so if Microsoft lost customers to System 10, they'd likely lose them forever.

Microsoft quickly planned for an ambitious release that was loaded with new features and performance improvements. It was tagged *SQL95,* borrowing the working moniker from the well-known upcoming Windows 95 release. Because the Big Question of 1994 was "What are your plans for replication?", replication became a keystone of the release. So did scrollable cursors, a feature that the developers had learned was necessary (from the ill-fated dBASE IV interface) to bridge the impedance mismatch between many record-oriented applications and a relational database. No mainstream DBMS product had yet provided a fully functional implementation of scrollable cursors in a client/server environment, and the SQL Server team believed that it was imperative to add this feature to the database engine. They had also been working on a dramatic new set of management tools, code-named *Starfighter* (today's SQL Server Enterprise Manager), which would also be included in the next release. The new feature list went on and on.

Microsoft's customers were clamoring to hear about the plans for SQL Server post-Sybase. So on June 14, 1994, Microsoft put together a briefing event in San Francisco for key customers, analysts, and the press. Jim Allchin, Microsoft senior vice president, walked the attendees through the plans for the future and for the SQL95 release. Attendees were impressed with the plans and direction, but many were openly skeptical of Microsoft's ability to deliver such an impressive product by the end of 1995.

NOTE

Some industry press began to sarcastically refer to the planned release as SQL97 and even SQL2000. Internally, the development team was still targeting the first half of 1995. To outsiders, they were more cautious—after all, this is software development, which is still more art than science. Stuff happens, so the SQL Server team said nothing more ambitious than 1995. (Everyone assumed that the planned date was December 31, 1995, and that they'd miss that date, too.) The rampant skepticism only served to motivate the SQL Server team even more to show that they could deliver. After all, the team rightly felt that it had *already* delivered an impressive release independent of Sybase. But no one gave them credit for that, so they'd just have to show everyone.

The team worked incredibly hard, even by Microsoft standards, to make this deadline and still deliver a full-featured, high-quality product. The first beta was released at the end of October 1994. Although Starfighter was not feature-complete yet, the database server was complete, and because the server takes the longest lead time for beta sites to really stress it, they went ahead with the beta release. This release was followed by a series of beta updates during the next several months, along with gradual beta-site expansion, eventually surpassing 2000 sites.

For nine months, dinner was delivered each night for those on the development team who were working late—usually a majority. On June 14, 1995, the product was released to manufacturing. Microsoft SQL Server 6.0 (SQL95) had shipped within the original internal target date, much sooner than nearly everyone outside the team had expected. It was an immediate hit. Positive reviews appeared in nearly all the trade publications; even more significant, none were negative or even neutral. Even more important than the press reviews was customer reaction, which was terrific.

InfoWorld, in its second annual survey of the 100 companies with the most innovative client/server applications in the previous year, showed Microsoft SQL Server as the number-two database. SQL Server jumped from 15 percent to 18 percent of those surveyed as the database server of choice—for a virtual tie with Oracle, which dropped from 24 percent to 19 percent. Sybase rose from 12 to 14 percent. Three of the top 10 applications highlighted by *InfoWorld* were built using Microsoft SQL Server.

Of course, the SQL Server development team was happy to see this data, but they took it with a grain of salt. Other data could be interpreted to suggest that Microsoft SQL Server's presence was substantially less than the surveys indicated. And the team recognized that they were still relative newcomers. From a sales perspective, Oracle was clearly king of the hill, and Sybase, Informix, and IBM were also formidable competitors in the DBMS market. Microsoft had not previously been a significant competitive threat. But now it was, and all companies armed their sales forces with bundles of information on tactics for selling against Microsoft SQL Server. Sybase, Informix, and Oracle all promised hot new releases. Although the SQL Server team had been sprinting for nearly four years, now was certainly not the time to get complacent.

Team-Building with Top Talent

Besides working hard on the development of version 6.0, Microsoft was also working to increase the size and strength of the team. It had built a small, crackerjack team that had delivered SQL Server for Windows NT, and this team was the core that delivered SQL95. But the team needed more people, more expertise, and exposure to broader ideas. So they went after top talent in the industry.

Microsoft attracted some industry luminaries—Jim Gray, Dave Lomet, and Phil Bernstein. They also attracted a lot of lesser-known but top development talent from throughout the industry. For example, DEC shopped its Rdb product around, looking to generate cash, and Oracle eventually spent a few hundred million dollars for it. But Microsoft didn't want to buy Rdb. Instead, it hired many of the Rdb project's best developers, augmenting this move by hiring several of the best recent masters' graduates who had specialized in databases.

The Next Version

After shipping version 6.0, many team members took well-deserved vacations after months of working in crunch mode. But within a month, they were actively beginning work on SQL Server version 6.5. With any huge release like version 6.0, some features get deferred due to schedule constraints. And during the ambitious 18-month project, new demands had come up that weren't even conceived of as requirements when the project began. For example, the Internet and data warehousing both exploded in importance and demand in 1995. Version 6.5 added capabilities for both. It also included further ease-of-use improvements, gained certification for conforming to the ANSI SQL standard, and provided much richer distributed transactions.

Although version 6.0 was released to manufacturing in June 1995, on December 15, 1995, Microsoft shipped a feature-complete beta version of 6.5 to 150 beta sites. The production version of 6.5 was released to manufacturing in April 1996, a scant 10 months after 6.0 was released. The SQL Server team was not about to slow down.

3 4

The Secret of the Sphinx

Slowing down was never really an option. Even before SQL Server 6.5 was released, an entirely separate team of developers was hard at work on a brand new vision for Microsoft SQL Server. In 1993, Microsoft had decided that databases would be a core technology in the product line, and in late 1994, it began hiring expertise from DEC and other major vendors to work with Microsoft's Jet and SQL Server teams to plan components for a whole new generation of database technology. During 1995, the period when SQL Server 6.0 and SQL Server 6.5 were being released, this team was building a new query processor that was planned as a component for what was to become the Microsoft Data Engine (MSDE).

Development of the MSDE was enhanced by the simultaneous development of OLE DB, which allowed elements of the SQL Server core product to be developed as independent components. Components could then communicate with each other using an OLE DB layer. At the end of 1995, the new query-processing component was integrated into the SQL Server code base, and development of a brand new SQL Server, with the code name *Sphinx*, began in earnest. The team working on the query processor joined the rest of the SQL Server development team.

The development of this new generation of SQL Server had one overriding theme: to re-architect the entire database engine so that the SQL Server product could scale as much as its users wanted it to. This would mean upward growth to take full advantage of more and faster processors and as much memory as the operating system could handle. This growth would also have to allow for new functionality to be added to any of the components so that, for example, the query processor code could add an entirely new join algorithm to its repertoire as easily as a new hard drive could be added. In addition to the upward growth, SQL Server would also be expanded outward to support whole new classes of database applications. It would also need to scale downward to run on smaller systems such as desktops and laptops.

There were two immediate goals for the first version of this new re-architected system:

- Full row-level locking with an extremely smart lock manager.
- A brand-new query processor that would allow such techniques as distributed heterogeneous queries and efficient processing of ad hoc queries. (Ad hoc queries are a fact of life when you deal with data warehousing, Internet-based applications, or both.)

This brand-new SQL Server version 7.0 debuted in a limited beta 1 release in the fall of 1997. Beta 2 was made available to several hundred sites in December 1997. Because of the new architecture, all databases and the data structures they contained needed to be completely rebuilt during the upgrade process. Microsoft and the SQL Server development team were absolutely committed to making sure all customers would be successful in moving from SQL Server 6.5 to SQL Server 7.0. A program called the 1K Challenge was instituted, in which 1000 customers were invited to send copies of their databases to the SQL Server development team to be upgraded to version 7.0. A porting lab was set up in the building on the Redmond campus where the entire development team worked. Each week from February to August of 1998, four or five ISVs sent a team of their own developers to Microsoft to spend an entire week in the lab, making sure their products would work out of the box with the new SQL Server 7.0. The core SQL Server development engineers made themselves available to immediately isolate and fix any bugs encountered and to spend time talking to the ISVs about the best ways to take advantage of all the new features of SQL Server 7.0 to make their own products even better.

In June 1998, Microsoft publicly released SQL Server 7.0 Beta 3 from its SQL Server Web site, along with a set of exercises demonstrating many of the new features and capabilities of the product. An Internet news server was set up so that any Beta 3 user could report bugs and ask questions of the SQL Server development team. More than 120,000 sites received the Beta 3 version of SQL Server 7.0. This included companies directly requesting the product through the Microsoft Web site, MSDN subscriptions, and Microsoft Beta Program participants (who receive beta versions of all Microsoft products when they become available). And then at last, after almost four years for some of the earliest team members, SQL Server 7.0 was publicly announced at COMDEX in Las Vegas on November 16, 1998. By the time of the launch, more than a dozen sites were in full production with SQL Server 7.0, including Microsoft's internal SAP implementation and barnesandnoble.com, HarperCollins, CBS Sportsline, Comcast Cellular, and Southwest Securities. The SQL Server 7.0 development team released the product to manufacturing on December 2, 1998, using build 7.00.623.07, with a code freeze date of November 27, 1998. The product became available to the public in January 1999.

Software for the New Century

As expected, the development team didn't stop working on SQL Server. Several planned features that had not made it into the SQL Server 7.0 release needed to be incorporated into the product. Other new features were already in the later stages of development in preparation for a version after 7.0. Two new code names were already in use: *Shiloh* was to be the version after 7.0, commonly assumed to be just a "dot" release like 7.5, and *Yukon* was the name given to the next major revision of SQL Server.

Initially, the SQL Server product managers were reluctant to predict widespread, immediate acceptance of SQL Server 7.0. Because it was basically a complete rewrite of the core engine code, many people might consider it a 1.0-level release. It was also anticipated that many potential adopters and upgraders would not be anxious to go to any release that ended in dot-zero and would want to wait for at least the first service pack. So initially, the Shiloh release was planned as just a "Super Service Pack." It would include perhaps a few new features that didn't get into 7.0 because of the shipping schedule, plus any bug fixes that were deemed too big for a normal, unnumbered service pack. The plan was to have a quick turnaround and release Shiloh no more than a year after the initial release of SQL Server 7.0.

Several factors came into play to change this initial vision of Shiloh. First, there seemed to be very little hesitation among the already installed user base to upgrade to the new SQL Server 7.0. Also, sales figures for new customers far exceeded even the most optimistic predictions. People were using SQL Server 7.0, and they liked it. Even after the public release of the product, Microsoft continued running the ISV lab in the building where the SQL Server developers worked, so feedback was ongoing and readily available to the development team. In general, the issues that came up as problems were easily fixable and a service pack for SQL Server 7.0 was released in May 1999. A second service pack was released in March 2000. There seemed to be no need for the "Super Service Pack" that Shiloh was intended to be.

A second factor that influenced a change in vision for the Shiloh release was customer requests. After referential integrity with cascading updates and deletes failed to make it into SQL Server 7.0, the product team heard loud and clear from the customer base that any further delay in adding this feature would not be tolerable. Customers also wanted better support for partitioned views and optimizer support for the star schema design used in data warehousing applications.

Finally, there was competitive pressure to make the next release bigger and better than originally planned. The Million Dollar Challenge issued by Larry Ellison of the Oracle Corporation pointed out a major piece of functionality that "they" had and "we" didn't. Adding materialized views to the product, which would allow SQL Server to meet this challenge, would be more than just a simple fix.

So the decision was made to have Shiloh be a full-blown release with at least an 18-month development cycle, but the version number would still be 7.5. The magnitude of the changes that would eventually find their way into this release was not immediately clear because the only absolute "must-have" at the outset was the cascading updates and deletes. It soon became clear that the release would grow beyond initial expectations. The team itself had grown substantially, spilling out of their original building on the main Microsoft campus into parts of two adjacent buildings. With a larger group of developers, a greater number of small to medium-size new features could be added to the product without seriously jeopardizing the planned release date. This book describes many of these new features.

In addition to having to add a substantial amount of new functionality, the development team also imposed upon themselves something they called "stretch goals." They declared an objective of a 20 percent increase in performance across the board, on all types of applications, but to give themselves something concrete to aim for, they extended this for specific applications. The main example of this was their goal to improve performance on the SAP R/3 Sales and Distribution benchmark by at least 40 percent. To do this, they had to make very specific changes in the optimizer that would directly affect the SAP queries but also benefit many other applications. At the Windows 2000 launch event in San Francisco on February 17, 2000, benchmark results were announced that demonstrated 6700 users for the Sales and Distribution benchmark, compared to 4500 users in SQL Server 7.0 for the same benchmark on the same hardware (an eight-way Pentium III-550). This represented a 48 percent improvement, and the goal had been well met.

After the development period was extended to a full 18 months, the decision was made to add another new feature. This decision was held in the strictest secrecy and wasn't even discussed with many high-level executives at Microsoft. The feature wasn't even addressed until after the release of Beta 1 in November 1999, and it was not announced publicly until the February launch of Windows 2000. Referred to internally by the code name *Coyote,* this secret project allowed SQL Server 2000 to support *distributed partitioned views* to allow unprecedented scalability in data management operations. It was this new feature that allowed the world-record-breaking benchmark results to be achieved and announced in San Francisco in February 2000.

Originally, these scalability features were scheduled for the version after Shiloh, but a close investigation revealed that many of the necessary components were already in place. These features included extensions to the optimizations of union views and the ability to update union views. Details of distributed partitioned views will be discussed later in the book.

The initial Beta 1 of Shiloh was released to a very small group of Early Adopters and serious beta testers in September 1999. Shortly after that, Microsoft announced that the official name of the product would be SQL Server 2000. There were two main reasons for this change. First, because of everything that was changing about the product, it was unrealistic to make it just a dot release (7.5); it really needed a whole new number. Second, if the product had gone with the name 8.0, it would be the only Microsoft BackOffice product that didn't use the name 2000. To conform to this companywide naming standard, the name SQL Server 2000 was announced. However, if you look at the internal version number using the *@@VERSION* function, you'll see that it returns the number 8.00.194.

It turns out that from the user's perspective, SQL Server 2000 introduces even more new functionality than did its immediate predecessor. SQL Server 7 had a completely rewritten engine, including brand-new storage structures, data access methods, locking technology, recovery algorithms, transaction log architecture, memory architecture, and optimizer. These, of course, are the things that this book describes in detail. But for the end user, developer, or part-time database administrator (DBA), the external changes and language enhancements in SQL Server 7 were minimal. SQL Server 2000 adds dozens of new language features as well as drastic changes to existing objects such as table constraints, views, and triggers that all developers and most DBAs will need to know about.

Because the internal engine changes are minimal, only two betas were planned. Beta 2, released in April 2000, was the widespread public beta and was sent to thousands of interested users, conference attendees, ISVs, and consultants. The SQL Server 2000 development team froze the code at build 8.00.194.01, on August 6, 2000, and released the product to manufacturing on August 9, 2000.

Internal SQL Server development is a never-ending story. Just as after the release of SQL Server 7, new capabilities were already well past the design stage, ready to be implemented. Features that didn't make it into SQL Server 2000, as well as continued improvements in scalability, availability, interoperability, and overall performance, continue to be addressed. The development team is using feedback from the beta forums and from early users of the released version of SQL Server 2000 to shape the future of Microsoft SQL Server.