

EmberWings

2025/2





The Firebird project was created at [SourceForge](#) on
July 31, 2000

This marked the beginning of Firebird's development as an open-source database based on the InterBase source code released by Borland.

Since then, Firebird's development has depended on voluntary funding from people and companies who benefit from its use.

Support Firebird

Thank you for your support!

EmberWings is a quarterly magazine published by the **Firebird Foundation z.s.**, free to the public after a 3-month delay. Regular [donors](#) get exclusive early access to every new edition upon release.

From Embers to Flight – 25 Years of Firebird

This June, as the days grow longer and the digital heat simmers, we find ourselves looking back—not out of nostalgia, but with admiration. Because next month, on **July 31, 2025**, the Firebird Project officially turns **25**.

A quarter-century ago, Firebird emerged from uncertainty, fueled by bold vision and open-source defiance. In the summer of 2000, when Borland released the InterBase 6.0 source code and corporate strategy wavered, a global community didn't flinch. Instead, it took that code, those ideals, and that opportunity—and turned it into something lasting. Something *alive*.

Firebird wasn't just a fork. It was a phoenix moment.

In this issue, we dive deep into those early days. Our feature article reconstructs the pivotal events that gave birth to Firebird—carefully compiled from published interviews, archived articles, and original press releases. It's a documented story of independence, ingenuity, and collective determination. This isn't just database history—it's a reminder of what communities can achieve when principles come first.

And Firebird hasn't just survived. It has matured. Evolved. Gained muscle. From embedded use cases to enterprise-class deployments, Firebird now serves countless systems around the globe—quietly, reliably, and powerfully.

Yet this moment is more than a retrospective. It's a milestone. A checkpoint. A call to the next generation: What will you build on the wings of this legacy?

In this issue, we also continue the hunt for Firebird easter eggs—yes, the quirky, clever secrets tucked into documentation, source, and software itself. Because even serious tech has a sense of humor—and Firebird's history has always been sprinkled with inside jokes and hidden gems.

So wherever you're reading this—from a dusty dev laptop in a startup garage to a high-availability cluster in a data center—take a moment. Firebird is 25 years strong. And it's not done flying.

Happy Anniversary, Firebird.

Here's to what's next.

*Warm regards,
The EmberWings Team*



In This Issue:

- Editorial
- Wisdom of the elders
- The Rise of the Phoenix
- Interview with Alex Peshkov
- Development update: 2025/Q2
- Toolbox: Firebird in VS Code
- Answers to your questions
- Planet Firebird
- Introduction to Saturnin
- ...And now for something completely different



The summer solstice arrived quietly, but the heat had been warning them for days. The air hung heavy in the office, despite the hum of tired fans and older machines.

The apprentice had not left his desk in over forty hours. His shirt was crumpled, his coffee stale, and his eyes locked on the Firebird console. Rows of performance logs scrolled like prayers in a forgotten language.

Master Lin entered without a sound. He paused, watching the apprentice dig through metadata like a monk deciphering ancient scripture.

"What are you fixing?" Lin asked.

"The performance tanked when I enabled the audit layer," the apprentice muttered. "Turns out the stored procedure calls another one that calls a third, and each one uses computed fields and external UDFs. The triggers fire on top of that. The call stack's a maze."

"Is anyone using it right now?" Lin asked.

"It's on staging," the apprentice said. "But if we don't tune it now, it'll choke in production."

Lin raised a hand. "You've been preparing since the moon was new."

He walked to the window and opened it wide. The air that came in was still warm, but alive – scented with jasmine and hot pavement.

"You know what today is?" Lin asked.

"The solstice," the apprentice said, not looking up. "Longest daylight. Useful for deployment."

Lin chuckled. "Not everything long is useful. Even daylight must know when to stop."

The apprentice rubbed his eyes. "But this is what it takes to master it, isn't it? Long hours. Precision. Control."

"Mastery," Lin said, "is not control. It is timing."

He pointed to the small server humming under the apprentice's desk. "Firebird does not care for your control. It runs by rhythm. Commit. Rest. Wait. Wake. It's closer to breath than battle."

The apprentice frowned. "But if I leave it now—"

"If you leave it now," Lin said, "you will return with clarity. If you stay, you will only sharpen your exhaustion."

He walked to the door, but paused.

"I will be by the river," he said. "There is shade. And cicadas. You're welcome to join me."

The apprentice looked back at the screen. The query plan blinked. Nothing urgent. Nothing broken. Just slow.

He closed the laptop. It felt heavier than usual.

Hours later, by the river, Lin skipped a stone across the water. The apprentice sat beside him in the grass, watching the sun tip slowly toward the horizon.

"I could bring my laptop here next time," he said.

Lin smiled. "You could. But then you would miss the way the wind changes before dusk."



The Rise of the Phoenix

By Pavel Císar

Once upon a time, there was a famous software company called Borland. In the 1980s and early 1990s, Borland was a heavyweight in the software world—an innovator responsible for Turbo Pascal, Delphi, and a suite of developer tools that shaped how programmers worked. Among the technologies it acquired and further developed was InterBase, a powerful, lightweight relational database that won respect for its low footprint, high performance, and easy deployment.

But by the mid-1990s, the software landscape had shifted, and Borland was struggling to keep up. The rise of Microsoft and internal missteps eroded its dominance. Strategic pivots failed to regain traction, and in 1999, the board brought in Dale Fuller as interim CEO to stabilize the company.

In December 1999, he made the decision to discontinue development of InterBase, though no official announcement followed. Unwilling to see the product quietly shelved, the InterBase division's management team resigned

in protest. When news of the decision leaked, it sparked concern and frustration within the user community, which quickly mobilized. From that momentum came the InterBase Developer Initiative (IBDI)—a grassroots effort to secure the database's future. Out of that pressure and passion emerged Firebird: a community-driven fork that carried forward the legacy of InterBase 6 and evolved into one of the most resilient open-source database engines of its time.

What follows is a reconstruction of those pivotal events, drawn from historic records, archived articles, press releases, developer notes, and direct quotes—piecing together how Firebird rose from the embers of InterBase.

INPRISE/BORLAND LEADS LINUX CHARGE: OPEN-SOURCES INTERBASE

Challenges other database vendors to follow its lead

COTTS VALLEY, Calif. - January 3, 2000 - Inprise Corporation (Nasdaq: INPR) today announced that it plans to jump to the forefront of the Linux database market by open-sourcing InterBase 6, the new version of its cross-platform SQL database. Inprise plans to release InterBase in open-source form for multiple platforms, including Linux, Windows NT, and Solaris.

"Inprise will be taking a leadership role in the Open-Source movement by releasing InterBase 6 under an open-source license. We are taking this bold step because we believe every Linux distribution needs InterBase," said Dale Fuller, interim CEO and president of Inprise Corporation. "By open-sourcing InterBase, we will be unleashing a world-class database for companies worldwide to develop and deploy business-critical, mobile computing, and Internet-based applications for multiple platforms, including Linux, Solaris, and Windows NT. This is an amazing opportunity for Inprise, its customers and the Open-Source community."

"Inprise open-sourcing InterBase is a step forward for software users," said Eric S. Raymond, president of the Open Source Initiative. "As open-source releases of operating systems and critical middleware become the norm, Inprise's release will undoubtedly raise the quality bar and customers will reap huge benefits in reliability, security, and total cost of ownership."

The source code for InterBase 6 is scheduled to be published during the first part of the year 2000. The company also announced it plans to continue to sell and support InterBase 5.6 through normal distribution channels. Inprise plans to announce further details of its roll-out plans for the InterBase open-source project on its Web site.

About InterBase 6

InterBase 6 is a powerful, high-performance cross-platform SQL database designed for business-critical, mobile computing and Internet-based applications on Linux, Windows NT, Solaris, and UNIX. Since 1985, InterBase has provided technologically superior relational database solutions to meet the business-critical database needs of companies like Motorola, Nokia, MCI, Northern Telecom, Bear Stearns, the Money Store,

the US Army, NASA, and Boeing. Through its ease of use, maintainability, simplified deployment and small footprint, InterBase has become the preferred embedded database solution.

About Inprise/Borland

Inprise Corporation is a leading provider of Internet-enabling software and services that reduce the complexity of application development for corporations and individual programmers. Inprise delivers integrated, scalable and secure solutions distinguished for their ease of use, performance and productivity. Committed to open platforms, Inprise continues its tradition of service and support for millions of software developers around the world through its online developer community and E-commerce site- <http://community.borland.com> - providing a range of technical information, value-added services and third-party products. Founded in 1983, Inprise is headquartered in Scotts Valley, California, with operations worldwide.

The First Major Open Source Database

by Doc Searls on Linux Journal, 4-Jan-2000

Doc Searls interviews Inprise President and CEO Dale Fuller.

Go to the Inprise (aka the Inprise Borland) Web site. There you will witness two highly significant developments: 1) the Corel-like linuxification of yet another leading PC software company; and 2) news that the company has open-sourced InterBase 6, the latest version of its cross-platform relational database. The product will be open source in all its versions, including Linux, Windows NT and Solaris.

InterBase is a first-rank relational database with customers that include Motorola, Nokia, MCI, Northern Telecom, Bear Stearns, the Money Store, the US Army, NASA, the Philadelphia Stock Exchange, First National Bank of Chicago and Boeing. It's hard to find an unkind word about this product, which competes with Sybase and Microsoft SQL Server, among other big-time database applications. So opening the source is a highly auspicious move.

Linux Journal senior editor Doc Searls talked about the move with Dale Fuller, Inprise President & CEO. Fuller came to Inprise less than a year ago to turn around what was once the premier franchise in software development tools - and in many ways still is. What followed has been a steady corporate drift toward Linux and the open source community, which is a natural constituency for Delphi and other popular Inprise Borland tools.

With this announcement, the company seems to have made a millennial shift into what might become a full-fledged Linux company (or at least as full-fledged as one can be when, like Corel, it still sells goods and services for other platforms). It also gives to the Linux community a very serious tools for migrating corporate databases off other platforms and onto Linux, among other significant things.

Doc Searls: I recently heard that you have some Linux credentials of your own.

Dale Fuller: Back in 1996 at WhoWhere, I developed the single largest Web site in the world running on Linux: Angelfire.com. It's still running on Linux and is one of the most popular sites on the Web.

Doc Searls: So why did you go with Linux?

Dale Fuller: When we did Mail City—an earlier site—we did it entirely in Solaris. It was roughly equal in size to

Angelfire, but Angelfire cost one tenth as much because we built it on Linux rather than Solaris.

Doc Searls: Today let's talk about what's going on with InterBase. Why did you go open source with it?

Dale Fuller: Until now, InterBase hasn't been a core asset at Inprise—at least as a revenue source. In that area it's been small. But it has huge value in so many other ways. This is a world class, bulletproof, fully-tested application that just about everyone needs. So to gain momentum in the marketplace we decided to open source the product. Open source is a word of mouth market, and that's what InterBase needs. And InterBase is what the open source development community needs, too, because InterBase not only fills a huge hole for enterprises, but gives them a way to migrate in a serious way from one platform to another - especially to Linux. There is an explosion of demand for Linux and we can serve that demand by providing a free and open database product that is already proven and tested over the last fifteen years.

Doc Searls: How big do you expect this to be?

Dale Fuller: We expect it to be huge. We have a gigantic following today, by pure market share standards. But what you want with open source is something that serves the whole community, not just a nice piece of it. It's a different game, and one to which this is both timely and well-suited.

Doc Searls: And the prospects were pretty bleak in the commercial marketplace.

Dale Fuller: Exactly. Having a great product isn't enough. You need to spend big marketing dollars just to begin driving awareness. We don't have that kind of money, and we're not interested in playing that game any more.

Doc Searls: Especially since word of mouth is free.

Dale Fuller: Exactly. The Linux market is still young, and our timing is still early, and the mechanisms for growth are extremely powerful. There is a huge need for what InterBase has already been doing for a long time, and the word of mouth in the open source development community should be very strong, very fast.

Doc Searls: And it will help pull through lots of other products.

Dale Fuller: Yes. That includes the Delphi product for Linux, the J Builder products for Linux. These will all be optimized to work very tightly with InterBase.

Doc Searls: What kind of license will you use? GPL? BSD? Mozilla? Something different?

Dale Fuller: We don't have a decision on that yet. But we do want to do right by the open source community. So let me just say at this point that we're thinking hard about it, and open to input.

Doc Searls: How does InterBase stack up against—or with—existing Linux database products, like MySQL?

Dale Fuller: It's complimentary. MySQL is nice as far as it goes, but InterBase is the only commercially developed, tested and deployed open sourced database product in the world. Its Java support is unbeatable. So is its performance. Even before we open-sourced it, no other RDBMS was easier to port between platforms. And no other database could scale from desktops to huge, industrial-strength systems. Those advantages will only increase.

Doc Searls: Are you going to manage this out of your current corporate infrastructure, or is this too different?

Dale Fuller: We are setting up a separate company to manage the whole process. We'll try to follow the paths already beaten by Red Hat, VA Linux, TurboLinux and others. We want to consolidate a core development team, make our money on service and support and adjunct products, do the compatibility testing, and work with the community to make sure there is one source that doesn't fork.

Doc Searls: Do you expect that database - the InterBase database - can be part of a standard Linux distribution?

Dale Fuller: I do. I absolutely do. We're talking about building material here. That's the great appeal of Linux: it's better building material, and open source is a better building method. To run with that metaphor, you need more than just nails. You need boards. You need lumber. In most enterprises, that's the data. And in most cases the data is more important than the applications. You need something that's bulletproof, that doesn't have memory leaks, that's reliable and proven. You get that with InterBase. It's a huge advantage.

Doc Searls: What's to stop others from offering the same thing?

Dale Fuller: Nothing. But they will not be as mature. Here's a product that is already on sixty different platforms, used by hundreds and hundreds of thousands of people. Our 911 system in the United States is run on it. Now it becomes available to everyone. Think about what that means for enterprises that are only beginning to adopt Linux now. It knocks down a huge barrier to adoption.

Doc Searls: You see it as a way for companies to migrate to Linux?

Dale Fuller: I see it in a big way. We built this as a cross-platform product. Any data that you have on Solaris will immediately run on Linux. You just port your application over. You don't have to worry about your data structures. That's fundamental to what makes InterBase such a sweet product. It runs on Solaris, Linux, Windows, AIX, multiple flavors of UNIX, even Wang systems from years gone by. So when you're an in-house company looking to expand and get out into the world of application server technology, you need a way for your data to move with you. The good news is, with InterBase, you can do this easily and quickly.

Doc Searls: Instead of building over again on another platform, you move whole buildings.

Dale Fuller: You move CAD drawings from one output to another output, rather than having to FedEx or fax them. You move the entire architecture over.

Doc Searls: Because the data structures remain intact.

Dale Fuller: Exactly. And because we have the fifteen years of experience at doing it successfully across all these platforms, we bring a lot to the open source table. Including big-time customers who need to find a way to make the leap to Linux. There is something for everybody here. We want to see people involved in testing it, improving it, submitting their findings and changes back to the organization, and circulating back out the constantly improved versions that anybody can use.

Doc Searls: And you want to build an organization customers can buy support from.

Dale Fuller: Yes. That's where we make our money.

Doc Searls: I think you're among the first in the open source world to directly go after what we might call Practical IT. These are guys getting work done in large organizations, busily adopting Linux because they like its virtues. They are part of the movement in a practical sense, but not part of The Cause. Most of them I would guess are not heavy Slashdot readers, or if they are they don't post there. Which makes them a bit invisible. But we know they are out there, because we know from IBM, HP and many other large companies tell us that their companies are full of these people, who are making the choice to migrate enthusiastically from NT and various UNIX flavors to Linux. It seems to me that's your constituency.

Dale Fuller: Yes. And they care about the future, too. Take a topic like EJB, Enterprise Java Beans. Those beans become transferable; I can sell, give, buy or take templates for, say, spreadsheets. This constituency likes that

kind of transferability across both vertical and horizontal platforms. Well, that's what open source InterBase will give them. No matter what platform you have, wherever you are, you don't have to worry about your data.

Doc Searls: So its virtues are a lot like Java.

Dale Fuller: Only not so closed. We're going open source all the way.

Doc Searls: And you think it will be easier to do that with a new company than with your old one.

Dale Fuller: Yes. What we're setting up will be the host organization outside our company. We don't want to do what Netscape tried to do, which wasn't very successful. We need a separate entity. We are funding it, but only as one source of funding. We want the community to help fund it, and own shares in it.

Doc Searls: Have you talked to some of the service players, like Linuxcare?

Dale Fuller: Yes we have. Also with Corel, TurboLinux, Red Hat and others.

Doc Searls: Are they ready to include you in their upcoming distributions?

Dale Fuller: We're moving to the next level of discussions. But I'm confident. This is the only serious database product in the world that is Linux compatible, that is actually available in open source form and that appeals at every level, in a very big way. There is no down side, for anybody.

Doc Searls: What does this do to Oracle?

Dale Fuller: It fires a bit of a cannonball across their bow. It is going to be much more difficult for them to follow along. Their whole game is about deployment and charging money after you develop on it. I think open sourcing InterBase will have a more immediate effect where Solaris and Windows NT are deployed. One reason is that it validates the Linux platform in a very significant way. It makes Linux far more competitive as a platform. With Linux you already have something that amazingly in a very short time is revolutionizing the world. This will step up that revolution by making databases a significant part of it.

Doc Searls: And in what situations does it threaten Solaris and NT?

Dale Fuller: Growing ones. NT is not the most scalable OS. Solaris is more scalable, but it's also expensive, so high cost itself is an impediment to growth. Customers are constantly looking for economically scalable solutions. The combination of Linux and InterBase gives them that. We also have a translator that transforms the entire data set out of SQL Server and into InterBase. And it will run under Linux or Solaris or any other platform. This is very appealing.

Doc Searls: The notion we've had for a long time is that you're building on an operating system platform. But in fact your building itself is mostly data, no matter what platform you build it on. Data is what you care about.

Dale Fuller: Yes. And what you get with an open source approach to data is far more safety. Because you can actually see what this database application is doing to your data. You can get under the floor and see what's causing the squeak.

Doc Searls: That's an interesting redefinition of data, because it presumes that responsibility belongs to the builders rather than the suppliers. It isn't like the old days when you wanted to trust the large company that would send out guys to fix your problem.

Dale Fuller: Enterprises increasingly want the flexibility to fix stuff themselves. Or have some choice about the help they get. You can still go out and hire somebody to fix your problem, but isn't it nice to know you have some options here?

Doc Searls: And that there is a wide-open market for those services.

Dale Fuller: I think what you're giving people is choices. Before they never had that.

Doc Searls: And you think scalability is the big issue.

Dale Fuller: I think scalability is going to be a gigantic problem, one that's going to make a lot of Linux converts out of a lot of big companies.

Doc Searls: One key is going to be exposing some of what's actually happening with Linux in enterprises. As we just said, that activity is not always visible.

Dale Fuller: In many cases, companies don't want to give away a technical advantage. It's a secret weapon. That's why I told my competitors at HotMail, "Man, we've got to stay Solaris all the time." Meanwhile we were marching down the path of Linux, making money because Linux was cheap.

Doc Searls: So you didn't talk up Linux when you were doing Angelfire.

Dale Fuller: No, we didn't. Another reason was that, frankly, we didn't hate Microsoft. In fact, Microsoft was willing to pay us millions of dollars to port one of our applications to NT. Maybe that put us out of the Linux mainstream at that time; but Linux has grown since then, and it now includes a lot of companies like ours, that use both platforms - and others as well. Companies like this take very pragmatic views of these matters. Basically, they want something that's bulletproof and scales. InterBase always gave them that, but now it will be in a new and better way.

Doc Searls: So you're clearly in the Linux movement now.

Dale Fuller: Back then we were deeply in the movement, but just not in a highly visible way. Now we're changing that.

Doc Searls: How does it feel to be a Linux company?

Dale Fuller: Great. I love what's happening now, because I was in the UNIX world a long time ago, in the early 80s, when AI was the rage. I worked on Common LISP at TI. The Micro Explorer was my project. The problem was, we had to do the sexy thing. We had to solve all the problems of the world, and that just wasn't a commercially workable goal. Industry kept coming back to us and saying, "Just solve one friggin' problem." We couldn't. Academia went for perfection and destroyed AI, because they didn't focus on the business aspects of it. I'm not seeing this with Linux. This time there is a consensus about the general direction of things, and solving business problems is a big part of it.

Doc Searls: As an ex-Apple guy, are you glad to see the company make open source moves with OS-X?

Dale Fuller: I think it's absolutely the right move. It will drive so much more adoption, and bringing in more ingenuity, more creativity. The more open you can make it, the more opportunities you have for developers to get in and actually make things worthwhile. That's the great thing about Linux. Nobody owns it, so it becomes a very creative platform. It makes me encourage Microsoft to open their OS.

Doc Searls: I don't think it's out of the question.

Dale Fuller: It's two years away.

Doc Searls: They need more of a sex change before they'll do it; but fundamentally they are a practical company, and they have no religion about keeping alive their dying products, which is a real advantage.

Dale Fuller: Meanwhile, there's lots to do here, and we're ready. And we invite Linux Journal readers to jump in and help with this thing, any way they can.

Addendum

When this interview was posted, the response was overwhelmingly positive. Some readers, however, urged us to give full credit to the other open source databases that were already out there and had prior claims to the "first major" label. The strongest urgings came from PostgreSQL developers, who kindly provided us with some points and links that we are happy to pass along here.

First, they point out that University Ingres, developed starting in 1977, qualifies for the "First Major Open Source Database" honor. Ingres is a direct ancestor of PostgreSQL. (Ingres II is now a product of Computer Associates.)

PostgreSQL is at version 6.5.3, and has had open source since the beginning. "The development is very open, the developers friendly, and the code is improving by leaps and bounds," writes Lamar Owen, RPM Package Maintainer with the PostgreSQL Global Development Group. He says "PostgreSQL has shipped with RedHat Linux as part of the 'Official Boxed Set' since RedHat 5.0." He also recommends comparing RDBMS by the "ACID criteria." These are: "Atomicity, Consistency, Isolation, and Durability."

Hacking database code is not lightweight work. "Kernel hacking is not a walk in the park, nor is GUI hacking, library hacking, or any other tool hacking," Owen says. "Database hacking is a league unto itself . . . the learning curve for doing back-end database development is the steepest of any project of which I am aware."

Inprise/Borland Forms New Company To Open-Source InterBase® 6

SCOTTS VALLEY, Calif. — February 14, 2000 — Inprise/Borland (Nasdaq: INPR) today announced the formation of a new company that will provide service, support and hosting for a new version of InterBase®, the first open-source embedded relational database product. The formation of the new company comes on the heels of Inprise/Borland's announcement last month to release InterBase 6, which is currently in beta, in open-source form for multiple platforms, including Linux, Windows NT and Solaris. The source code for InterBase 6 is scheduled to be available mid-2000.

The executive team of the new company, called InterBase, will be led by Ann Harrison as president, Paul Beach as vice president of sales and marketing, and Jim Starkey, the founding architect of the original InterBase product, will serve as the technology software architectural advisor for InterBase.

InterBase is a high-performance, standard query language (SQL) database that is well known for its low-maintenance, enterprise-class relational database management system (RDMS). Designed for business-critical, mobile computing and Internet-based applications on Linux, Windows NT, Solaris, and UNIX, InterBase supports many high-profile customers such as Nokia, Société Generale and Deutsche Telekom.

"We plan to finance InterBase with funding from the announced Inprise/Borland Venture Fund, announced last month, as well as several of our venture capital partners," said Dale Fuller, interim president and CEO of Inprise/Borland. "By combining the InterBase technology with an experienced management team in a start-up environment, we are enabling users to leverage the power of this world-class database for multiple platforms, including Linux, Solaris, and Windows NT. We believe the open-source process will greatly increase the distribution of InterBase in the marketplace."

"The Internet economy demands database-driven applications," said Harrison. "InterBase has always been a fast, low-hassle database. As an open-source product it will move out of its current niche and become a force in the new economy."

As president, Ann Harrison is responsible for helping InterBase develop and implement a strategic plan for achieving substantial long-term growth in the open-source market. Harrison brings over 15 years of database experience to the new evolution of InterBase 6, and will be responsible for a wide array of technology and business initiatives for the company. With Jim Starkey and Don DePalma, she formed the original company that developed the innovative database architecture that became InterBase.

Prior to his appointment to VP of sales and marketing for InterBase, Paul Beach was the general manager at Inprise/Borland for the InterBase product. Based in the United Kingdom, Beach was responsible for sales, marketing and business development activities for the product. During his tenure at Inprise/Borland, Beach increased revenues of InterBase by US\$4 million annually.

The source code for InterBase 6 is currently scheduled to be available mid-2000 and will be open-sourced under the MPL 1.1 model.

Disappearing Inprise spawns InterBase

As featured in the 451 (www.the451.com) on April 10, 2000 By Janos Gereben

San Francisco -Even as Inprise/Borland is becoming part of Canada's Corel (while fighting IONA Technologies in court), it is spinning off a new company, InterBase Software Corporation (IBSC) to provide what it calls "the first open-source embedded relational database" – InterBase 6.0.

Shown at the Software Development 2000 in San Jose, InterBase – still in beta, and due for release this summer – runs on multiple platforms, including Windows NT, Solaris, and the focus of Corel's development work: Linux. Such companies as Nokia, Societe Generale, and Deutsche Telekom will use current and future versions of InterBase.

James Starkey, who will be architectural advisor for the new spin-off, designed the original InterBase, launched in 1984 and later acquired by Ashton Tate. Ann Harrison (coming from the original developer of the database architecture which became InterBase) will be president, Paul Beach (former general manager of InterBase at Inprise) is vice president of sales and marketing.

Harrison considers the timing for the move into the Linux market to be "exactly at the right moment. Linux is going to emerge as a considerable force in the commercial market with proprietary applications built on open source foundations. One foundation for applications that handle data is a database. We're going to be there," she said.

IBSC's entrance is into a shrinking database market for smaller applications. Microsoft's Access – and, less, FoxPro – dominates, with Sybase and Informix hanging in there. New development work is going on in Europe, especially Germany. InterBase, predictably, is trying to position its product in-between: "We're a big step above Access and FoxPro," Harrison says, "in terms of our ability to handle multi-user highly concurrent applications."

Size, of course, has always been relative, and the issue is becoming even more blurred with faster and more powerful PCs. Historically, the mini computer was the master at controlling peripherals (controlled, in turn, by

mainframes), but it also managed masses of data. As of now, what was on minis is now on PCs and, in some cases, what was on mainframes is now on PC networks, but Unix, Netware and NT run the networks.

Beach sees an increase in small-database applications: "With the appearance of fully functional relational databases at the low end, such as InterBase, Watcom [now Anywhere,] SQL from Sybase, and Personal Oracle, many developers have realized the benefits of replacing their Paradox, dBase, Access and similar applications with real database applications, whether single-user or multi-user. We have many customers who develop small applications which can run independently on a laptop, for example, but scale to full servers supporting hundreds of users."

Given proper scalability and portability, open source databases, according to Beach, can outclass the dBASE-Paradox-Access structure, which "lack functionality and run out of steam." Beach also takes issue with the jaundiced view of small-scale database applications: "If this is a dead area, why did Microsoft announce a cut-down version of SQL Server when it launched SQL Server 7.0 as a replacement for Access? Microsoft may not be the fastest moving company in the world, but they are not stupid," Beach said.

A key condition for InterBase to be successful with IB 6.0 – or any other business in the field – is to find a niche in the enterprise model: large, distributed databases served on a network connected via LAN, WAN and Internet to clients with controlled access.

Forming An InterBase Company

By Charles Babcock, Inter@ctive Week, March 27, 2000

InterBase is a mature relational database management system that is slated to become open source code in the second quarter under a new company, InterBase Software. Some observers might ask, however, why, if it is so mature, it hasn't succeeded as a commercial database system.

The answer, said Ann Harrison, president of the new company being spun out of InterBase's current owner, Inprise, is that it has never been the focus of an organization dedicated to advancing it.

Harrison joked that the new company should be named InterBase Software the Fourth, because InterBase started as an independent company, which was acquired by Ashton-Tate, which in turn was acquired by Borland International. Borland of course became Inprise/Borland, which is now merging with Corel. InterBase Software, including about a dozen existing Inprise employees, is expected to become an independent company before the merger is completed, she said.

So far, InterBase has always been behind when it comes to having a powerhouse company to market it. It started out with \$253 in capital to purchase the first employee's chair, "Sybase was starting out at the same time with \$20 million in capital," Harrison noted. "When it was sold to Ashton-Tate, that company was collapsing," she added.

Ashton-Tate was sold to Borland, but as a tools company, Borland never knew how to sell database technology," Harrison said. A tools company sells its products to developers, but developers don't make the purchase decision on an enterprise relational database system, she added.

In addition, James Starkey, her husband, designed special features into InterBase when he first conceived of a relational database system that would be easy to use and maintain. Indexing, restoring data integrity after a failed transaction and other administrative operations are simpler through InterBase by design, she said. Starkey was the designer of Rdb, the relational database system sold by the former Digital Equipment, and

crafted the lessons he learned from the experience into InterBase. He will serve as a technical advisor to the new company but will not become an employee.

One of InterBase's basic features is that it is a multiversioning database system. As a series of transactions are completed or an upload of data is finished, InterBase automatically preserves the old version of the database alongside the new version. When an automatic cleanup mechanism in the database comes across a transaction that didn't finish, it knows to retrace the transaction steps, restoring data from the earlier version so that data integrity is maintained, Harrison explained.

The multiversioning approach makes InterBase a simpler system to administer in several ways: backup and restore of the database may be executed as processing continues, Harrison noted.

There is some extra overhead associated with a multiversioning system, making it less likely to be selected for high-traffic tasks. "Our performance is adequate for up to 200 concurrent users," Harrison said, but as a free, open source system that's easy to administer, she figures many Internet start-ups would be happy with a reliable database system capable of managing those numbers.

Inprise/Borland and Corel terminate proposed merger

SCOTTS VALLEY, Calif., — May 16, 2000 — Inprise/Borland Corporation (Nasdaq: INPR) today announced that its merger agreement with Corel Corporation (Nasdaq: CORL, TSE: COR) has been terminated by mutual agreement of the two companies without payment of any termination fees. The reciprocal stock option agreements have also been terminated.

Dale Fuller, Inprise/Borland interim president and CEO said,

"Much has changed since the merger was agreed to more than three months ago, and our board concluded that it would be best to cancel the merger on an amicable basis."

Commenting on Inprise/Borland's future, Fuller said,

"Inprise/Borland is well positioned today with improving operating results and substantial liquid assets. Future operations will continue to follow our increasingly successful strategy of creating solutions that enable companies to move their businesses to the Internet. We will create the tools necessary to leverage new platforms for Linux, Solaris, and Windows 2000, and provide superior services to Application Service Providers. In addition, Inprise/Borland will continue to invest in companies and technologies that complement this strategy."

In January of 2000, Inprise/Borland and Corel entered into a confidentiality agreement that included a standard three-year standstill covenant. That agreement remains in effect.

Interbase And Kylix Details From Borland/Inprise Con

As featured on Slashdot on July 12, 2000

"I'm typing this from a machine in the computer lab at the Inprise/Borland conference in San Diego, where many new details about Interbase and Kylix have been revealed."

"The conference opened with a "Matrix" themed intro, and there have been blue and red jellybeans all over the place. The major announcement at the intro was that JBuilder will be ported to Mac OS X, with full support for Aqua. The press release is [here](#).

Paul Beach, Interbase VP of marketing and sales, gave a very informative talk. It's very clear that he has a clue. Many of his talking points were interchangeable with what Bob Young would say, asked similar questions. He even alluded to the tired old Heinz ketchup analogy to explain brand equity. Interbase's plan is to sell support contracts and box sets, just like the other big open source companies. Paul is very realistic and down-to-earth about what they expect to happen. I think they are going to do very well indeed.

The most startling revelation, though, was that Interbase 6.0 is done and ready to ship the manuals and CDs are duplicated and printed and sitting in boxes. He even brought a box of CDs to the talk. Reading between the lines, it seems like these CDs have been sitting around for some time. But Dale Fuller (Inprise CEO), who also showed up at the Interbase product address, will not permit the CDs to be distributed until all the contracts are signed and executed and appropriately lawyered. Dale promised, both at the opening keynote and at the Interbase talk, that the contracts would be done and the product would ship within two weeks. Here's hoping he holds himself accountable to that promise.

Many details about Kylix have been revealed. Kylix is installed on several machines in the lab, but unfortunately these machines are off the net, so you can't steal a copy (darnit). The IDE was demoed at one of the talks, but it wasn't in the lab. What was on the lab machines was the command-line Delphi compiler only. It seems quite solid. I played around with it, built some projects that displayed various forms and controls, that sort of thing. As expected, the compiler is blindingly fast, builds genuine native executables, and looks very similar to Delphi on Windows. The new class libraries, called CLX, are very nearly 100% source-compatible with the Delphi VCL. The CLX visual components are wrappers around QT, and will be available on Windows as well in Delphi 6, so if you code to CLX you will be cross-platform. Kylix builds fully compliant QT apps, with KDE-style theming and all the bells and whistles. As with all QT/KDE apps, you can run a Kylix-developed app under Gnome but it isn't seamless for purposes of theming or UI consistency.

Kylix will, of course, be closed-source, closed-process, proprietary, and will have a price notably divergent from zero. The initial release will be Delphi standard and professional, followed quite quickly by C++ standard and professional. The enterprise version of Kylix will be called "Kylix Studio" (or similar) and will include both compilers in a single SKU. Someone in the audience suggested that they do a combined SKU under Windows as well, which got a lot of applause. At the opening keynote, Dale said that a major goal for Kylix is to make it possible for developers to release their own projects under any license, including full-strength GPL. This is a worthy objective, but I'm not sure Borland truly understands the ramifications. Then again, I'm not sure I do myself. What would it mean to have a GPL project, but you have to use a proprietary compiler to build it? Presumably the compiler libraries will be proprietary, and you have to link with them if you want a usable binary. This bears thinking about.

For database access from Kylix, there's a new library called dbDirect. Or perhaps it's called dbExpress. And I think it might also be called DataCLX, or perhaps DataCLX is a superset including dbDirect plus other things.

Anyway, whatever it's called, it's a new library that lets applications talk to databases in a consistent way just like the BDE. But unlike the BDE, it tries to be as 'thin' as possible, bringing the application code as close as feasible to the native database vendor APIs while still providing relatively good code portability. With the BDE, Borland has to write a complex driver for every new database they support. With dbDirect, it's a simple matter of wrapping a few vendor API calls. Application developers can mix and match calls to dbDirect with calls to native database APIs. dbDirect will also have a tiny footprint compared to the BDE, including the ability to statically link right into the application. DataCLX will be the only database API in Kylix, and will ship in Delphi 6 side by side with the traditional BDE setup.

Kylix will also include a web broker architecture, very similar to what's in Delphi 5 today. Where Delphi web modules can compile to traditional CGIs or ISAPIs, Kylix web modules will compile to traditional CGIs or Apache modules. The level of integration between Kylix and Apache is impressive. This part of things is called NetCLX and also includes the usual socket components and TCP-suite stuff like telnet, smtp, etc.

In addition to the Borland announcements, there are a few dozen vendors on the trade show floor. Big names include Sun, Caldera, Linuxcare and Cobalt. Corel is pointedly absent. And of course we have the usual gang of third-party component vendors people like TurboPower, Woll2Woll, Raize, Digial Metaphors, etc. Everyone seems to be planning to port their tools to Linux, promising release dates from a couple weeks to a couple months after Kylix ships.

The bottom line is, I'll be coming home empty-handed. No Kylix beta, no Interbase source. But I have a very strong sense that the Delphi community is gathering behind Kylix in a big way, and I'm very pleased to see Interbase poised on the verge of release. Just get those contracts signed, Dale!"

Inprise/Borland introduces InterBase® 6.0 Now free and open source on Linux®, Windows™ and Solaris®

SCOTTS VALLEY, Calif., — July 16, 2000 — Inprise/Borland (Nasdaq: INPR) today announced the availability of the source code for InterBase® 6.0, its cross-platform, structured query language (SQL) relational database management system (RDMS). Binary formats for the Linux®, Windows™ and Solaris® operating systems are also available for download, free of charge. With Version 6.0, Inprise/Borland introduces performance improvements and a higher level of SQL92 compliance.

Developers worldwide have built InterBase into a multitude of applications, taking advantage of InterBase's small footprint and low maintenance requirements. Copies of the source code and binary versions for Linux, Windows, and Solaris are available at the Inprise/Borland Web site, www.inprise.com/interbase/.

"This is a great day for Inprise/Borland," said Dale Fuller, interim president and CEO of Inprise/Borland. "Combining the energy of the open source market and the power of InterBase will create an explosion of open source applications. Even among commercial application developers, a reliable, low-hassle, free database is big news."

InterBase 6.0 has been released under a variant on the Mozilla Public License (MPL) V1.1. Developers using InterBase under this license can modify the code or develop applications without being required to open source them. The open source license applies to all platforms.

InterBase 6.0 is the most recent version of the powerful, high-performance cross-platform database known for its low-maintenance requirements and enterprise-class features and performance. In addition to being the first open source release of InterBase, Version 6.0 introduces a number of new features including new data types (long integer, date, time, timestamp), extended SQL92 compliance, performance and security enhancements.

Availability and Support

InterBase 6.0 can be downloaded immediately from the Inprise/Borland Web site at www.inprise.com/interbase/.

Will Borland let InterBase go?

As featured in the 451 (www.the451.com) on July 21, 2000 By Janos Gereben

San Francisco - The next few weeks will be crucial to the survival of InterBase Software, the database spin-off from Inprise Borland. Or at least the hopeful spin-off because, so far, Interbase hasn't yet managed to separate itself from its parent, and consequently hasn't been able to turn its code over to the open source community, as it said it would at the start of this year.

InterBase's community of developers and customers, including recently-signed OEM Cobalt Networks, are getting frustrated. They are hoping to hear Inprise's interim CEO Dale Fuller agree to set InterBase free at Monday's annual shareholders meeting. If not, insiders say that some of the key members of the InterBase team may just walk away from the whole project. Official sources aren't ready to comment yet, although they say that Inprise is putting together an official announcement for this Wednesday.

Why does InterBase matter? Advocates say there's a real need now for a solid open source database for Linux. The current contenders, MySQL and PostgreSQL, aren't commercial-grade products, while InterBase has been on the market since 1985 and runs on NT and Solaris, as well as Linux. It's also got the right characteristics - easy to maintain, simplified deployment, small footprint - for use as an embedded database. In other words, it's a database that can be integrated within other applications - invisible itself, but acting as a crucial underpinning.

However, if the original development team breaks apart and Inprise itself loses interest, InterBase will enter the growing category of open source abandonware. Without a dedicated recourse to push development and marketing, it will go the same way as the other open source databases.

Some history is needed. Originally an independent company, InterBase was acquired over the course of a few years by Ashton Tate, which was itself swallowed up by Borland in the early '90s. As a high-end database, it always fitted somewhat uneasily at Borland among the PC-oriented Xbase database products such as Visual dBase (now open source itself). Boeing, Motorola, Nokia and the US army are among its largest users.

When Inprise/Borland decided to exit the database market in 1999, InterBase looked doomed. And at the end of the year, five of the key development tools quit the company, having been told that research and development budgets were to be cut. Within the course of a month, staff at the database unit is said to have dwindled from 40 to around five. But then Fuller responded to suggestions that InterBase be released as open source. He announced the intention to do so in January this year, challenging other database companies to do the same. The move encouraged Ann Harrison, codeveloper of the original InterBase, along with founding architect Jim Starkey, to take on the role of president of a proposed new InterBase spin-off. Starkey

agreed to act as software advisor, and Paul Beach re-joined as VP of sales and marketing. The team began getting the next release, InterBase 6, ready for shipment, and also started working on a new set of deals to help jumpstart the company's relaunch.

By June, all should have been in place. But according to insiders, legal delays in sorting out the separation have held things up. Two crucial dates were missed. The first was the recent Borland Developer's Conference, where 5,000 CDs of InterBase 6 were ready for distribution, but couldn't be released. The second was Cobalt's launch of the RaQ 4r, which uses InterBase as its default system. Frantic, last-minute negotiations allowed the launch to go ahead, but Cobalt only has agreements to use the system for 180 days - further use beyond then is dependent on InterBase becoming a separate entity, say sources.

It's not clear if it's just a long-winded legal process or something else that has held things up. At the end of June, Fuller and Harrison issued a joint statement on the situation: "The reason for the delay was neither code cleanup nor any lack of direction or effort on anyone's part. Simply put, separating one division from a company is a difficult process." Separately, Fuller assured the451 that an announcement was due "any day."

But what will the nature of the announcement be? The spin-off was originally looking to acquire the licenses, trademarks, copyrights and patents of InterBase for a sum thought to be around \$6m, payable over two years. Inprise was offering a \$2m investment from its internal venture fund. The core InterBase team would be re-hired before they disperse. But if a separation can't be negotiated, then the InterBase code might be turned over to the open source community under the Inprise banner. Without an internal champion within Inprise, that's unlikely to be an effective strategy.

Inprise/Borland Terminates Negotiations to Sell Interbase® Product Line

Announces Plans to Support Interbase Community

Friday July 28, 9:42 pm Eastern Time

SCOTTS VALLEY, Calif., July 28 /PRNewswire/ — Inprise/Borland (Nasdaq: INPR - news) today announced that it has terminated negotiations with Ann Harrison regarding the sale of the InterBase® product line to a start-up venture led by Ms. Harrison. The company is planning to fully support InterBase customers by retaining a core team to successfully transition the open source version of InterBase. Inprise/Borland is responding to the needs of open source developers, and is joining the many companies that have embraced open source.

"After careful consideration, we determined that it was not in the best interest of Inprise/Borland's stockholders for us to sell the InterBase product line to a start-up entity which initially would be dependent on Inprise/Borland for funding," said Dale Fuller, interim president and CEO of Inprise/Borland. "We are excited about building a relationship with the open source community. We believe this process will strengthen both the InterBase community and further enhance the product."

Last week, Inprise/Borland announced that the source code for InterBase 6.0 and binary versions for Linux®, Windows®, and Solaris™ would be made available at the Inprise/Borland Web site, www.inprise.com/InterBase/. InterBase 6.0 was released under a variant on the Mozilla Public License (MPL) V1.1. Developers using InterBase under this license can modify the code or develop applications without being required to open source them. The open source license applies to all platforms.

Monday 31th July, 2000

The InterBase 6.0 code released by Borland under IDPL license was forked as project [firebird](#) on SourceForge.

InterBase Saga Continues

by Michael Bernstein, published on TECHNOCRAT.NET on August 16, 2000

The nascent open-source database project that was coalescing around the Inprise/Borland InterBase product has run into some problems.

Recap:

Back in February, Inprise/Borland (hereafter referred to as Inprise) announced that the InterBase RDBMS was going to be released as open-source, and that its maintenance was going to be spun off into a separate corporation (the Interbase Software Corporation, or ISC). After some delays, Inprise finally released the source on July 25th.

So far, so good.

Three days later, on July 28th, Inprise announced that they were in fact keeping InterBase for themselves, basically hanging the developers working for ISC out to dry.

Specifically, Although Inprise has released the InterBase source code, Inprise is no longer investing in ISC or transferring the underlying copyrights and trademarks to ISC.

The reason given at the time and since has been that the proposed deal with ISC (which included ISC accepting a \$6 million debt note, and Inprise buying a 19% stake in ISC for \$2 million, as well as a number of other tangible and intangible assets) "when considered in its entirety, was not in the best interests of our shareholders". This in spite of the fact that Inprise's move to 'sunset' InterBase in December was the impetus for the formation of the original IBDI (or InterBase Developer Initiative) that later became the ISC. Most of the current 'value' in this product that Inprise was going to write off, is inherent in the community that has grown around it, especially now that the source has been released.

Inprise declined to elaborate on the details of the negotiations, citing an agreement with Ann Harrison not to do so (Ann Harrison was not available for comment due to currently being on a sailboat off the coast of Maine), but claimed that ISC had created conditions under which Inprise was not able to agree to proceed with the planned investment.

There has been some speculation that the deal fell through due to the downturn in technology stocks in general and Linux and other open source stocks in particular, so the short term prospect of Inprise recouping their investment wasn't too promising, or that the ongoing income from InterBase was too critical to Inprise meeting its quarterly financial statements, but none of these could be confirmed.

Inprise's announcement, made just before a weekend, had sent the InterBase mailing lists into a frenzy of speculation and acrimony aimed at Inprise. Most of this has now died down, as developers have started to redirect their efforts taking Inprise's actions into account.

While there was some concern at the time of the initial release that not all of the source code had been made available, Inprise has since added most, if not all, of the missing code modules. At this time, although the latest source code and CVS tree have been released, some key pieces are still being held by Inprise. These include:

- release notes
- build instructions and makefiles
- build documentation including known bugs list.
- the testing suite
- the InterBase name

A key issue for the community is the ownership of the documentation. Inprise is the copyright holder to the original documentation, But ISC had commissioned an outside company to update the documentation in anticipation of being transferred the copyright. According to Paul Beach, who was appointed VP of sales and Marketing at ISC, the new documentation is not 'derivative' enough (over 30% different) to allow them to release the new version without violating Inprise's copyright.

Another issue is the 'test suite', a set of databases with which to do conformance, performance and stability testing of the database software. Strictly speaking, the test suite contains no source code, so an open content license would probably be most appropriate. ISC had intended to release the suite after they had the opportunity to modularize it and make the files more usable to the individual developer. Ted Shelton, Senior VP of Business Development at Inprise, has said that "The reason we have retained the testing suite is so that we can evaluate putting a licensing program in place so that ISC and other companies that want to build InterBase related businesses will have this revenue stream option. Releasing to open source is a one-way journey. We need to make sure that we aren't destroying a needed revenue source for ISC and other potential InterBase companies before we can make a decision about open sourcing the test suite."

This would probably mean that individual developers would have to either pay a company to QA their code, or else wait for someone else to discover any bugs in their modifications.

This also points out what some have seen as a difference in perspective; while communicating with people involved with the ISC, the sentiments most often expressed are a desire do what is necessary to help developers self organize to improve the code, while when communicating with Inprise, it seems as though anything smaller than another corporation doesn't even register, and developers are expected to help Inprise.

Helen Borrie, moderator for the InterBase mailing lists, said "Ted Shelton has not so far shown any willingness to discuss the coal-face requirements. I still hope that he can be persuaded to discuss it. It is my belief that he doesn't understand how/why individual developers and teams would need to use [the test suites]."

Ted Shelton commented on one of the InterBase mailing lists, "We are not shutting down InterBase. We continue to have a solid group of InterBase employees, we will continue to support our existing customers, and we will continue to participate in the InterBase community. We are working to develop a plan around supporting InterBase as an open source product. We will be inviting the community to make suggestions about how we can best accomplish this goal. While we were not able to come to an investment agreement with Ann Harrison and Paul Beach we appreciate the efforts they have made on behalf of the InterBase community and wish them the best in their endeavors."

This statement, while undoubtedly seen as conciliatory and reassuring by Inprise, provoked a very hostile response in the open-source InterBase community, including accusations that Inprise had set an unrealistically high price for the hand-over. A figure of \$10 million was quoted by several people on the lists but this figure seems to have been an exaggerated mis-extrapolation of the \$2 million/19.5% numbers.

Craig Leonardi posted a calmer than average response "Given your company's long tradition of neglecting Interbase, I am amazed at your press release. I can only imagine that all the Sr. VP's drew straws to see who would be the author While I do believe that you would welcome new companies to support Interbase, your assertions that Interbase will not be shutting down, that you are working on a plan, and that you appreciate the efforts of Mr. Beach and Ms. Harrison, fly in the face of reason and fact."

The list moderator, Helen Borrie, stated that "The community objected to Ted Shelton's approach - asking the community to help Inprise - when it had already spent six months putting its own infrastructure in place. The sincere Open Source approach should have been 'How can WE help YOU?'"

The 'solid group' that Ted Shelton refers to had 40 developers prior to December, which is when most of them left, leaving less than ten at Inprise. According to Paul Beach, a former General Manager of the InterBase Business Unit at Inprise, those who remained, did so because they thought that the spin-off would happen. Since the failure of the deal, three Inprise InterBase employees have resigned, and it's uncertain how many will remain with the company. Certainly two, since they rely on Inprise for their Green Cards.

As for participating in the community, Inprise is not seen as having any of the structures in place to adequately support the community's efforts, and few, if any, seem to be willing to give their assistance to Inprise. One indication of this are the two projects on SourceForge: The Inprise sponsored project has seven developers and administrators, most of whom seem to be Inprise employees (and one of whom is Mr. Shelton), while the 'Firebird' project, led by Mark O'Donohue, has twenty. Also, other projects are sprouting up around porting and tools efforts for various platforms.

Interestingly, the name change from 'InterBase' to 'Firebird', while suggestive of a fork, is actually due to Inprise's request that the InterBase trademark not be used, since trademarks must be defended or they are considered to be 'abandoned'. This has also prompted those involved in the ISC to start referring to themselves as 'NewCo', in lieu of a new name being selected.

It's interesting to note that while the IPL grants Inprise a privileged position WRT releasing binary-only versions of the database and can include community contributed code in those releases, they are not the copyright holders of any community contributed code. So, while they can keep their version of the software in sync with the community version and reap disproportionate rewards for a while, eventually the entire codebase will have been replaced, and the software could be re-released under a different license such as the GPL, removing their privileged position for any versions produced from that point on, and also removing the possibility that anyone else would be in a privileged position thereafter (including NewCo).

Looked at this way, open source licensing is just a formal way of stating that the only asset that any company or project has is the people involved in it.

While Inprise has certainly not done anything illegal, since they are in fact the copyright holders of the InterBase source code, they seem to be committing what might be called a 'reverse fork' (a term I just coined). They had, for all intents and purposes, already transferred maintainership to the NewCo group in a very public way. According to open source 'community standards', maintainership is a matter of consensus, and the way that consensus was subsequently ignored by Inprise has prompted the community to move forward with their own agenda, and shun Inprise as if they were the interloper. To many observers, it seems as though the only way to avoid a fork now is for Inprise to acknowledge the community as the de-facto maintainer.

The community consensus seems to be that most of the key InterBase developers already do not work for Inprise (at least 5 former Inprise InterBase engineers are active in the community effort), and that a fork, while painful without the pieces that Inprise has not released, is now probable but not yet inevitable.

The NewCo steering committee is currently exploring alternative business models and soliciting companies for indications of interest in support contracts, so that they can move forward with plans that do not depend on any Inprise involvement, and there is every indication that they will succeed, as current InterBase customers, many of whom have been disillusioned by Inprise's failure to allocate marketing and development resources to InterBase, are showing intense interest in a community supported version of the database.

Inprise, for its part, says that it has not yet ruled out other investments in outside InterBase companies.

An Open Letter to the InterBase Community from Ann Harrison

Published on August 23, 2000 on ibphoenix.com website

Who are we ?

The IBPhoenix group. For now, the site is a static collection of articles, frequently asked questions, and links to all the known resources for InterBase, but a number of the articles here, have never been published elsewhere. IBPhoenix is the source for documentation of the design intentions of InterBase as well as conversion information, usage guides, and community news.

What is the IBPhoenix group?

Well, for one, there's me. I'm Ann Harrison and I've been involved with InterBase from its inception fifteen years ago. For another, there's Paul Beach, whom many of you know as the last general manager of the InterBase Business Unit at Borland/Inprise. His history with the product may not be as long as mine, but his involvement has been intense. In addition, Helen Borrie, Bill Karwin and Pavel Císar have been instrumental in getting the IBPhoenix site up and running.

Why have we formed IBPhoenix?

Because we believe that the InterBase community wants and needs reliable support - both for usage and for design issues. Thousands of companies around the world depend on InterBase. They need services and IBPhoenix is here to supply those services.

What is the relationship between IBPhoenix and Borland Software Corporation (BSC)?

None. Inprise released the InterBase sources on July 25th. IBPhoenix is working with the community to improve and extend that code base. We are not affiliated with Borland in any way.

Is this a fork in the code?

The InterBase code is currently available from Source Forge as InterBase and as Firebird. We believe that a single source tree is best for the InterBase community and will work to reunite the two trees. At the same time we consider Firebird to be the real open source project - one where developers cooperate and learn from each other. In short, I'm off being didactic on the Firebird lists.

I hope the IBPhoenix site will be useful to you and that you will visit it often. New information is appearing regularly - daily - as we mine the decades of experience we and the community bring to InterBase.

IBPhoenix - where is it going ?

The question should probably be "IBPhoenix - where are we going?"

For the long term, I can only give you my answer, which is "As far as we can." InterBase is a very serviceable

database. It fits well into web applications. It fits well into a variety of data intensive applications. It could be better. That's the direction IBPhoenix is going: making InterBase better, more robust, faster, more secure, easier to use, easier to learn, and better integrated with the tools developers use. And we will go as far in that direction as our skills and resources allow. Not to own InterBase, not to control its development, but to work with the community to make it better.

In the short term, what's going on with IBPhoenix?

Most of you know that the recent history of IBPhoenix and of InterBase has been eventful. In its former life as ISC, IBPhoenix had one set of plans and expectations. Over the past month, we have been scrambling to replace the facilities, tools, and funding that were part of our expectations. We have a small, but very competent group available to offer support services. We're involved with Firebird, the more active of the open source InterBase projects. We've asked to be involved in Inprise's open source project, in hopes of avoiding a fork that would damage everyone's reputation and reduce InterBase's chances of success. We've built a web site with a vast amount of information for InterBase users and developers. Our web site changes daily.

Over the next few weeks or so, we will make the web site interactive, adding threaded discussions of bugs and projects, searchable archives of various InterBase lists, and an incident tracking system. We will continue to publish articles about known issues with InterBase, with suggestions for avoiding them. Our goal is to make InterBase better, not to hide the defects.

In the slightly longer term, we're adding e-commerce: selling services and reselling companion products to InterBase. That effort is longer term because changing the name of a corporation has amazing ramifications when dealing with banks and secure network services. When the bureaucracy is defeated, the IBPhoenix site will be the source of open source software, shareware, trustware, and commercial tools for the InterBase community.

Who is this "InterBase community" I'm talking about? You are. Organizations that use InterBase V4 on AIX or VMS or DG/UX or any of the other platforms are the community. Companies, large and small, that depend on InterBase are the community. The developers at Firebird are the community - members who have created working versions of InterBase on BSD and AIX, and who are turning the sight of a hundred eyes on the code and making it better. Inprise is part of the community. At the heart of the community is a group called the InterBase Developers Initiative, IBDI. They run the web site for InterBase 2000 and they made all this happen.

They're still making it happen. Today, David Robinson, released a new InterBase V6.0 kit for Windows, based on the same material, but without the packaging problems that affected the Inprise release.

Thank you David, I'm proud to be associated with you.

Ann

InterBase Flap Could Fork Code

By Maureen O'Gara as featured in Client Server NEWS August 25, 2000

Inprise or Borland or whatever it calls itself these days was supposed to open source InterBase and then turn it over to a company that InterBase's original developers were putting together.

It got as far as open sourcing the code for the end-of-life database that in its heyday was an Oracle wannabe. Oh, okay, Sybase wannabe.

The company also open sourced the Linux, Windows and Solaris binaries for InterBase 6.0, the latest rev, same rendition as the source code it released.

Everything is out under a variant of the Mozilla license, which means developers don't have to open source their modifications or InterBase6-based applications. Mozilla makes InterBase different from the MySQL database, for instance, which is going GPL, because GPL would require MySQL apps to be open sourced. (See www.inprise.com/interbase).

That was the week before last. At the end of last week Inprise publicly pulled the plug on the negotiations it had been having with its InterBase general manager Ann Harrison and the projected InterBase Systems (ISC) spin-off since at least St Valentine's Day. Harrison, who walked out of Inprise when the negotiations hit the wall, was one of the original gang of four that created InterBase. She is married to Jim Starkey, the original founder of InterBase, who sold the company after seven years to Ashton Tate. Starkey, who by his own admission gave up recommending InterBase a year and a half ago because of the "gross incompetence of Borland management," was to serve as ISC's technology software architecture advisor. Inprise, by the way, formally announced ISC's formation in mid-February.

Late last Friday, Inprise's interim president and CEO Dale Fuller said in a canned statement that, "After careful consideration, we determined that it was not in the best interest of Inprise/Borland's stockholders for us to sell the InterBase product line to a start-up entity which initially would be dependent on Inprise/Borland for funding." Inprise said it was going to support the open source effort itself instead and was retaining a core transition team.

Inprise had originally said it would underwrite ISC from its venture capital fund.

Inprise senior VP of business development Ted Shelton said in a series of increasingly revealing but not completely satisfying e-mails to developers, many of whom feel Inprise acted in bad faith, that, "we were not able to come to an investment agreement with Ann Harrison and Paul Beach," ISC's VP of sales and marketing and that "we made offers to Ann that she did not accept, and that Ann made offers to Inprise that we did not accept."

In a more expansive message on Sunday, Shelton said, "Ann and ISC proposed to have an independent company from Inprise and asked for the technology, trademarks, and for a venture capital investment. There were, in addition, a number of legal requirements that they placed around the transfer of these funds and the intellectual property. A significant effort was made to reach a middle ground in our negotiations with Ann. However, we were not able to bridge the differences in opinion. As we have an obligation to our shareholders and employees in addition to our obligation to our customers, we could not simply agree to ANY conditions that a third party was going to put on a transaction of this kind." Inprise had originally conceived ISC as a company to service, support and host InterBase 6. At least that's the way it described it in its St Valentine's Day press release.

Shelton also said that Inprise is "working on a plan around supporting InterBase as an open source product" and that Inprise would "welcome the development of new companies seeking to support InterBase."

Fuller has made a lot of people unhappy with his decision and not the least of them are Harrison and Starkey, who might very well go ahead and form ISC anyway and fork InterBase.

Starkey and Harrison, who on Friday, after the axe fell, continued to describe herself as president of "NewCo," the re-dubbed ISC. disappeared at the weekend - at least they're incommunicado and not answering e-mail -

and are evidentially considering their options. The InterBase community appears ready to follow them down the NewCo path, judging from the postings to the public mailing lists. NewCo may be restyled Firebird and InterBase or rather the InterBase clone given another name and tag line.

Starkey in mid-July blamed Inprise for a lack of resolve in open sourcing InterBase and getting ISC off the ground. "Deal after deal has been lost," he said of the delays. "Potential investors have been turned away. Important customers have drifted away as Borland's lawyers missed deadline after deadline."

Even without Harrison and Starkey, who have been with the product for 20 years, the community could fork the database. Inprise didn't release the InterBase 6 documentation - written partly by Inprise and partly by ISC - cross-platform tests, build scripts or bug lists. (It's rumored Inprise wanted ISC to pay \$10 million for the stuff.) The community could surely reproduce them and fork the source. Some enterprising folks have already figured out how to re-build it, at least on Linux. There are notions about hosting any schismatic effort on VA Linux Systems' SourceForge site. Anyway, there are moves to set up a replacement web site, get the source tree organized, put documentation together and synchronize the code.

Inprise has also alienated its VAR community, which first started to drift when Inprise decided to kill InterBase off in December - an idea it recanted when word leaked and a tempest ensued. Then there are the people who have been working for free on little items like drivers under the impression that they'd get a piece of ISC.

Angry developers says Inprise released "unbuildable source code, broken binaries and [is] withholding the pieces that members of the Open Source community need to actually give this product a life."

Meanwhile, Inprise went ahead Monday and announced an InterBase 6 deal with Cobalt Networks that was shaky a couple of weeks before. Cobalt is going to bundle the relational database with its RaQ 4r server appliance. Cobalt imagines InterBase applications will be "appliantize'd" and pre-configured with its Linux-based platform. InterBase 6 is largely cast as an embedded RDBMS since it failed as an Oracle contender.

Paul Beach takes credit for putting the Cobalt deal together, an 18- month effort at relationship-building. Inprise was supposed to release InterBase 6 binaries to Cobalt by July 14 but Fuller, who had reportedly promised to release the code, for some reason turned the issue over to his lawyers at the last minute. They refused permission for the code to be released. At least that was the story given out.

At the time, Starkey sent around an e-mail saying, "Unless Borland agrees to release version 6 binaries to Cobalt today, July 14, the Cobalt deal is probably dead. If the Cobalt deal dies, the Borland/InterBase deal may die with it. In any case, if the Borland/InterBase deal does not close by July 24, the deal will be dead. The future of the product will be Dale Fuller, not Ann, Paul and Matt. I will not be involved in any capacity." Well, Beach apparently salvaged the Cobalt deal and Starkey was right about the InterBase arrangement.

InterBase 6 introduces new features including new data types (long integer, data and time), extended SQL92 compliance, an open interface for defining new national character sets plus performance and security enhancements.



The SQL query published in the last issue, when executed over the EMPLOYEE sample database, proves that the famous **LOST numbers** can indeed be found everywhere.

Whats Happening to InterBase? - Summary of the year 2000

As featured in the UK Borland Users Group Magazine (Nov/Dec 2000) By Paul Reeves (BUG InterBase Group Leader)

A Recap

Just before Christmas 1999 Dale Fuller's management team had a meeting with the senior executives of the InterBase division and told them of their new business plan. It has been stated by informed personnel that they had decided that the revenue stream was insufficient for the size of the division. They also said they wanted to lay off at least 50% of the current InterBase staff of 39 (including part-timers and contractors). In addition, there would be no budget to restore the struggling US marketing and sales staff, which had already been reduced to one 3/4 time person and an intern. This was not a workable scenario to grow a product business. It has never been clear to me just what Dale and his team had in mind, but I believe it was intended to cancel the release of 6.0 and sit on the diminishing revenue stream from 5.n for a couple of years.

Possibly unbeknown to the Inprise management team, the InterBase team had been quietly building a sound and profitable (i.e., not loss making) business. One of the original developers of InterBase (Ann Harrison) had continued to provide excellent technical support for the product via the mers list server/news group. The product manager, Bill Karwin had equally spent many hours there helping developers. On the business side Paul Beach, responsible for worldwide sales (non-US) was basically generating about 90% of the total turnover. With little resources he had concentrated on building a formidable network of VARS that typically integrated InterBase with other Borland tools.

Their response (Karwin and Beach) to the new business plan was probably not what was expected. They resigned. Their resignations hit developers and VARS hard. Contractually they couldn't explain why they resigned, but gradually the above picture became clear. Thus the InterBase community was born. Suddenly we discovered that some serious business and serious money was riding on InterBase. Enough people came forward to buy InterBase several times over. A sale was not to be, but the development community was now forming itself into an organised force to be recognised.

Initial Public Offerings

Autumn '99 saw the Linux Bubble on the stock market. Companies associated with Linux and Open Source were IPO'ing like frenzy and seeing huge (paper?) profits on stock value. Dale Fuller's primary duty is to his shareholders and he worked hard to raise the share price. The development of Kylix for the Linux platform was widely touted. The venerable Borland C command line compiler was made available for free. And suddenly he had his top InterBase executives suggesting they Open Source InterBase if he wasn't going to fund further development. I guess he thought about this idea over Christmas and, come New Year, announced that InterBase would indeed be made Open Source. It was also announced that a new company would be formed to manage this. Inprise would hold some 20% of the stock and would help find venture capital to fund a large part of the rest.

Even the bitterest opponents had to admit that this was a smart PR move, if nothing else. It helped the Inprise stock price remain steady, maintaining a half yearly average of well above 10 dollars. (Previously the stock had been languishing, to put it mildly.) The consolidation of the stock price helped cement a good deal for the Corel merger.

In February 2000 it was finally announced that Ann Harrison and Paul Beach would run a new InterBase

Management company. The original architect of InterBase, Jim Starkey, was also on board as chief technical advisor. The source would be made available by end of June at the latest and the company spun off as soon as the legal stuff was sorted.

After the black days of last December it looked as though Inprise had finally got things right. An Open Source InterBase was warmly accepted by just about everyone, in and out of the InterBase community. Ann Harrison was unchallenged as the right person to act as the figurehead for the new company. Paul Beach knew the InterBase business and, if anyone could make the open source project work money-wise, he could. Having Starkey back on the team was the icing on the cake. The synergy of a highly respected open source database shipping with every copy of Delphi, CPPBuilder, Kylix and JBuilder looked a winner.

The Bubble Bursts

During the year, it seems as if some of the assumptions Inprise made have changed. The Linux/dot.com stock market bubble started to burst. Then, the Corel deal came apart, much to the relief of many on the Borland side at least. Subsequently it became clear that IPOs were no longer a ticket to instant riches.

With the prospect of a hugely profitable IPO receding far into the distance, it appears as if the Inprise management decided that they should find another way to leverage their InterBase asset. It seems as though the original intention had been to live off the licence and maintenance fee income for the next three years, so they offered the new company a chance to pay this money upfront in one hit by charging them for assets that had not previously been valued as part of the deal. Rumours went around that \$10 million was the asking price for such things as the documentation and the test suites. This rather changed the deal, which by that point had been all but signed.

Although the deal collapsed Inprise had still made a public commitment to open source InterBase. The bad PR that would arise from backtracking on that promise would have been huge, so they had little choice but to follow through. Thus the source was released, albeit a month later than promised.

Whatever the reason for the deal failing, it does appear that Inprise has rather shot itself in the foot. By spurning the new InterBase Company in this way, it has engendered the unremitting enmity of the InterBase development community. About 99% of them use Borland tools. Inprise are also likely to alienate some of the Linux development community too. Linux is very close to its Open Source roots and Inprise appear intent on writing the how-not-to guide for commercial companies wishing to climb on the Open Source bandwagon.

Inprise have issued a public statement saying that they intend to continue supporting and developing InterBase. It is difficult to take this seriously, given the events of the last year. The strength of this commitment is evident from the release of the source itself. Several bits were missing and they were released in piecemeal fashion over subsequent days. It was several weeks before all the parts were available in a single download. Worse still, the source would not compile. No serious work had been done to ensure that InterBase could be built outside of Scotts Valley. Things have improved slightly, but not at the speed of the Internet.

What's Happening Now

Within days of InterBase going open source, the InterBase development community had created the Firebird project on [sourceforge](http://sourceforge.net). Developers quickly got to grips with the task of fixing all the broken bits in the install routines. Since then, work has started on understanding how InterBase works internally. This will take a while but is a necessary pre-requisite to actually making enhancements or adding new features to the product. Work has also started on creating new ports of InterBase 6.0. Meanwhile, successful builds have been done for FreeBSD and Mac OS X, for instance. Inprise have finally released the core of the InterBase test suite, so it is now possible to certify new builds.

The new company that was not to be, has now formed as [IBPhoenix](#). Their website evolves so quickly that it is hard to keep pace. It really is the best starting point for all resources concerning IB. They have most of the best InterBase experts available to them and are offering support contracts for companies that wish to purchase them. They are also working closely with the FireBird project to ensure the future development of InterBase.

Interest in and usage of InterBase seems as strong as ever. The events of recent months have done nothing to diminish the range of third party stuff available. There is too much to list here but all the links can be found at the IBPhoenix web site. In addition to a raft of management tools (all built in Delphi) there is support for running IB with PERL, Python and PHP. There are two new ODBC drivers available for InterBase 6.0 and development of an ADO driver is underway.

The community around InterBase continues to grow. At the last count there were some fourteen sites in the web-ring, covering three languages. Perhaps the best starting point is the InterBase developer initiative at <http://www.interbase2000.org>. It has many additional resources provided by developers.

It is difficult to predict how things will develop from here. For now, we shall just have to be content with an industrial strength database that is open source and freely deployable. All else aside, this is a major step forward. Let's hope it is not the last.





Interview with Alex Peshkov

Alex Peshkov is a long-time core developer on the Firebird database project, with a background in systems programming. He's held critical roles on the team including security coordinator, buildmaster, and ports authority—titles that hint at both his deep technical expertise and the behind-the-scenes work that keeps the Firebird engine running smoothly across platforms. His experience spans decades of evolution in the project, starting from its roots as an Interbase fork to today's actively maintained, feature-rich open-source database.

In this interview, Alex shares insights into his work and the thinking behind some of Firebird's most important architectural developments. From system security and cross-platform compatibility to metadata caching and the introduction of new data types, readers will get a closer look at how complex database features come to life—and the hands-on problem solving it takes to make them work reliably in the real world.

Can you tell us a bit about yourself, your background, and how you first became involved with the Firebird project?

Born in 1963 in Yaroslavl (USSR). Married, have two children, already grown up. Studied at the Moscow Institute of Physics and Technology from 1980 to 1986. Started working with computer equipment in 1984, at the research institute where I did an internship (at that time it was a Soviet clone of IBM/360, the PL1 language). And I continue to work in this field today. At the end of 1999, the company where I worked decided to use Interbase as a database server, not least because an open source release was promised. But Interbase lacked the ability to execute a dynamically generated (in a PSQL procedure) SQL statement, and for the current project this was very important. I decided to implement it myself - it was IB-6.01. Now, of course, I am horrified when I remember that code - but surprisingly it worked! Since sharing useful changes in open source is at least good form, I sent the modified files to some Interbase Ring - and forgot about it for about a year. And a year later I received a letter from Dmitry Emanov with an invitation to join the project.

Can you share a memorable moment from your early days working on Firebird that shaped your commitment to the project?

A moment - definitely not, it was an interesting job from the most beginning.

You have been the “security coordinator, buildmaster, and ports authority” of the Firebird project since the very beginning. Everyone has some idea about security, but building and ports are rather mysterious areas for most users. Could you tell us a little about what that job entails, and what challenges you had to overcome? What Firebird ports have you worked on, and what is Firebird’s future outlook for supporting different systems?

There is nothing mysterious in these terms, and in some sense they mean the same thing. By the word "port" in this context we mean the ability to compile Firebird for a specific system, including in the word "system" both the OS and the hardware and the compiler. In this case, as a rule, it is implied that such a system is not included in the list of the main ones on which Firebird is supported. By the way, this list of main systems can change, and has changed more than once during the

existence of the project. Especially considering that Firebird was a fork of Interbase, which in turn became the development of the Groton Database System. (Hence gds at the beginning of some API calls.) For example, at one time there was a port for Windows 3.1, which has now become one of the 2 main supported systems. Ports live and die - usually due to the obsolescence of hardware. 20 years ago a 64-bit port for amd64 (x86_64) has just arrived, and now support for 32-bit builds is mainly for compatibility with old client software.

The port consists of 2 main parts - changes in the source code in C++ and provision of the automatic build process for a given compiler and OS. So it is quite difficult to separate these 2 types of work. Although of course sometimes changes in the build are also required when developing the system architecture, for example, before restoring the providers architecture in FB3, each source module might be compiled up to 3 times in different modes - classic, superserver and superclient. Providers eliminated the need for this, which certainly required reworking the Firebird build process.

You've been instrumental in enhancing Firebird's security, from fixing vulnerabilities to introducing advanced features like database encryption and SQL-driven User Management. What motivated you to tackle this area?

At the first step - grant suggested to me by FF specifically for security development. And later - the work is interesting, I wanted to deal with this further.

Speaking of Firebird security, you started with mapping of Windows security to Firebird 2.1, that was further enhanced in version 2.5. How this work influenced the design of new security architecture introduced in Firebird 3?

Before Firebird 3, it was hard to call it mapping. The idea of what a full-fledged mapping should be like arose precisely during the work on Windows trusted authentication, and its first version (in private builds) was implemented for Firebird 1.5. But to fully implement this idea, plugins, providers, and a reworking of the authentication system as a whole were needed. So, it is unlikely that 2.x had an impact on the new security architecture introduced in Firebird 3. Rather, what you could see in 2.5 was a stub of this new security architecture - so stubbornly cut that the remains could be implemented in 2.5. By the way, since then I realized that

for temporary switches, temporary API calls, etc., you can't be lazy about specifying them in the documentation - this feature will be immediately removed after a full implementation of such and such a feature in the future. Keeping compatibility with these temporary solutions over several versions was labor-intensive and, most importantly, essentially useless.

You left a big mark on Firebird 3, as your signature is under many ground-breaking features, like providers architecture, plugins, interfaces and new object-oriented API, database encryption, over-the-wire compression and completely new security architecture. What are some of the biggest challenges you've encountered while developing them, and how does the database's architecture influence your approach to solving them?

Probably the most difficult thing was that most of the listed changes are very dependent on each other - it was impossible (without a significant increase in the volume of intermediate work) to implement them sequentially. Providers architecture, plugins, interfaces, new object-oriented API and new security architecture had to be implemented (at least as a first approximation) in one commit. And to say that it was a big commit is to say nothing. Add to this that at that time we had just switched from CVS to SVN, GIT with its branches and merges was still a long way off, i.e. I could not use a lot of wonderful GIT features that time. After that, encryption and compression of data transmitted over the wire fell into the new architecture softly and naturally. Database encryption stands apart. The issue of monitoring the state of the database and changing it was resolved quite effectively and gracefully (background encryption with restart from a checkpoint in the event of a server restart). But the problem of storing and transmitting the encryption key is still waiting for its real solution

Firebird 4 introduced further improvements to system privileges and cryptography (backup encryption and built-in cryptographic functions). Do you consider this area feature-complete now, or do you plan to introduce additional enhancements?

In future versions of Firebird, it is necessary (as was already mentioned) to centralize control and transfer of secret keys. The idea is not yet fully thought out - but a secure storage of such keys is needed, which could be worked with directly

from SQL. For example, when accessing another server from PSQL, not all the keys known in the session should be transferred, but only one, specifically needed. And of course, it is necessary to support an up-to-date set of encryption algorithms - for example, the RSA we have now is rapidly becoming obsolete.

In Firebird 4, you've also added support for DECFLOAT, INT128, and increased precision for exact numeric types. What challenges did this pose — particularly with cross-platform and cross-processor compatibility, given the limited native support and frequent need for software emulation — and how are you addressing them?

Adding any new data type creates the first problem - lack of support for it in client software. (Please note - it has nothing to do with cross-processor compatibility!) Even the simplest BOOLEAN generated a lot of problems, since several fields of this type were added to the system tables and a significant number of software had problems accessing that system tables. Imagine how that problem grows with new types of digital data with internal format unknown to most of hardware (except some PPC). New datatypes were added primarily to improve the accuracy of calculation results in Firebird and the latter means that even as a result of operations with old data types, the server can return new ones. I will give an older example - since the maximum number of records in the table exceeded 2^{32} , we began to use BIGINT (64 bits) as the result of the count() function. But some large companies for a long time (maybe someone still does?) continued to use special builds of Firebird in which count() returned 32 bits.

For this reason, the SET BIND operator and the associated DPB tag and configuration parameter were added, which set the rules for transforming the request's metadata at the server level before returning it to the user. I think that the presence of this feature has simplified the transition to new versions of Firebird for many users, and this operator also very well fits one more new datatype - time with time zone support. As for cross-processor compatibility - probably the most difficult problem was to find a big-endian box for testing and debugging. The Fedora project helped a lot with it, they release packages for S390x including Firebird. There were no significant problems with the code - basically the libraries used for working with such types of data coped perfectly. Only the network level required minor improvements based on the debugging results -

transfer of messages with new types of data in a machine-independent form.

Support for batch insert and update operations in the API was another feature you developed for Firebird 4, and it was also used to enhance the performance of gbak restore operation. However, many drivers still does not provide support for this feature, mostly because of the complexity of BLOB handling, and somewhat limited and "obscure" documentation. Do you plan to enhance the examples/documentation to help driver developers to better understand how to implement the batch support in all its possible variations?

First, we need to clarify which drivers have problems with this. I am sure that those that use fbclient to talk to Firebird server have enough information on how to implement work with BLOBs. Moreover, no one forces them to use the most difficult option - independent formation of a data stream from several BLOB fields. The existing calls addBlob (create a BLOB and place the first portion of data in it) and appendBlobData (add another portion of data to the current BLOB) do not present any difficulty and are discussed in the examples.

Another question is those drivers that work directly with the wire protocol. fbclient forms a stream from the data received by it, which is transmitted to the server in fairly large blocks, which actually determines the improvement in the speed of work. That same stream should be formed by such drivers. Format of this stream is really non-trivial, especially due to the fact that the segment BLOB contains the lengths of the segments, which may require conversion (BIG / little endian) when transmitted over the wire. Add to this alignment and packing into blocks of a certain size - yes, I spent some effort to debug this in my time.

And here comes the second point - the network protocol has never (before the document prepared by Mark) been documented anywhere, except for the source texts themselves. Yes, the data stream in batch is a kind of add-on above the level of packets used by drivers. But I made additions to the document describing the format of data exchange over the wire and I can't quite imagine how else to document the format of this stream. Although in my opinion the best documentation in this is the source code, fortunately we work in the open source sphere. Also I am always ready to answer specific questions about what is happening "here and there".

You're currently working on the shared metadata cache for Firebird 6. What is this feature, why is it a game-changer for users, and what challenges are you facing in its development?

Any cache is necessary to improve performance, and metadata cache is no exception. You probably know about the page cache, which stores the most actively used pages of the database file. This is a general-purpose cache, everyone uses it. But some data (specifically, data from system tables, also called metadata) is needed by the Firebird server especially often, primarily for compiling and optimizing queries. Metadata cache is used to speed up these operations. The cache is very efficient - the required record is extracted from an array (units of assembler commands) instead of searching for it in a table (even taking into account index search, an incomparably longer operation).

But the metadata cache that has existed since the times of Interbase (and even earlier) was oriented towards exclusive access to it. For completely understandable reasons, this was enough for a single-threaded classic server. The superserver (before FB3), although multi-threaded, had one active thread - the others waited for it. In 2.5 SS had one active thread per database, but since each database provided its own metadata cache - exclusive access requirement was satisfied.

When designing FB3, it was decided (in order to speed up the already protracted release) to have one copy of the metadata cache per attachment. This provided exclusive access, but at the same time (as in the superclassic) N copies (practically identical) of metadata are stored in one process. This is not so bad while the metadata cache is small. But in some cases (primarily this is the implementation of business logic using SQL in triggers and procedures), the volume of the metadata cache becomes not so small, the value of 1Gb is quite achievable. On the one hand, it is not much today, on the other hand, the number of attachments today can also be quite large. Even with 1000 attachments, the consumption of 1Tb of RAM for storing 1000 identical copies of data looks, to put it mildly, questionable.

In addition, the existing metadata cache is not sufficiently correct and stable. I will not give examples of errors, because (especially before 3.0) they could get >9 CVE points. And the most unpleasant thing is that fixing these errors without fundamentally reworking the metadata cache code is impossible.

Due to the above, it was decided to implement a shared metadata cache in FB6. The main difference is that it does not try to do it's best to keep single, correct for everyone copy of metadata object. Instead permanent header is stored in an array and a number of versions (transaction-dependent) are linked to it. What for challenges - probably the main problem was the need for constant reverse engineering of the old code to understand what should happen here and there.

Looking ahead, what is your vision for Firebird, and what do you hope to achieve in your future work on the project?

I'm unlikely to answer the first half - maybe something very general like "achieve compliance with the SQL standard", I deal with technical issues, vision is not my area. And there are many plans for development. In addition to Batch, I would like to see Pipe in the API - a similar Batch but 2-way interface, using which the client sends a stream of requests to the server and receives a stream of responses to them. Naturally, it requires certain improvements in the server architecture, first of all - centralized thread management. Also, as I already noted, secret key storage is needed.

And perhaps I will not look further and try to guess.

Thanks for you time!





Development update: 2025/Q2

A regular overview of new developments and releases in Firebird Project

Releases:

- [firebird-butler-protobuf 2.0.0](#), released 30.4.2025
- [firebird-lib 2.0.0](#), released 30.4.2025
- [fdb 2.0.3](#) - Legacy Python driver for Firebird 2.5, released 3.5.2025
- [firebird-driver for Python 2.0.2](#), released 21.5.2025
- [firebird-uuid 1.0.0](#), released 28.5.2025
- [firebird-uuid 1.0.0](#), released 28.5.2025
- [Jaybird 6.0.2](#), released 31.5.2025
- [Jaybird 5.0.8](#), released 31.5.2025
- [firebird-base 2.0.2](#), released 2.6.2025
- [saturnin 0.10.0](#), released 2.6.2025
- [saturnin-core 0.10.0](#), released 2.6.2025
- [saturnin-sdk 0.10.0](#), released 2.6.2025
- [firebird-qa 0.20.2](#), released 8.6.2025

Firebird introduces support for Windows ARM64 platforms

The master branch was updated with support for Windows ARM64 builds. The changes cover updates to build scripts, configuration files, and Visual Studio solution/project files to accommodate ARM64 architecture, ensuring compatibility and enabling compilation and functionality on Windows ARM64 platforms. The ARM64 build are available among other daily [master branch snapshots](#).

Mass UPDATE problem solved

When updating a table with 172 million rows where only non-key integer columns were changed, Firebird still executed referential integrity (RI) triggers for every foreign key referencing that table. With 1,358 foreign keys, this led to about 1e11 trigger invocations—turning an update that should have taken minutes into a 19-hour ordeal

The core issue is that Firebird's engine indiscriminately fires RI triggers on updates of tables with foreign key relationships—even when primary/unique keys aren't altered. Thus, every update, regardless of relevance to key fields, led to trigger executions.

A change was made via PR #8600 to skip RI trigger invocation when no primary or unique key has been modified. After applying the fix in a test build, the same update ran in just 13.5 minutes—a ~85% performance improvement from the original 19 hours.

Support for Spatial Data Type

On April 4, a new proposal was presented to the Firebird developers.

This proposal introduces spatial data type support in Firebird, enabling the database to handle geometric and geographic data. It aligns with standards from OGC and SQL/MM, implementing types like POINT, LINESTRING, and POLYGON, with optional dimensions (Z, M, ZM). These can be specified using GEOMETRY or GEOGRAPHY based on the spatial reference system (SRS) used—

Cartesian or geodetic, respectively. Syntax for creating fields, tables, and spatial reference systems (SRID-based) is clearly defined.

Spatial objects can be represented using standard text (WKT/EWKT) or binary (WKB/EWKB) formats, with functions provided to convert between formats and manipulate spatial data. The system supports constructors and analytical operations such as `ST_Intersects`, `ST_Buffer`, and `ST_Length`. While most OGC and SQL/MM functions are implemented, support for advanced types like curves and surfaces is still pending.

Internally, spatial data is stored in BLOBs using an EWKB format. Metadata is tracked in system tables like `RDB$SPATIAL_FIELDS` and `RDB$SPATIAL_REF_SYS`, with provisions for user-defined systems. The implementation leverages third-party libraries like `Boost.Geometry` for geometric operations and `PROJ` for coordinate transformations, with performance optimized via caching of computed SRS characteristics.

Indexing strategies are still under evaluation, and integration of system views to comply with OGC is experimental. The proposal is robust, laying the groundwork for high-performance spatial data processing in Firebird, while remaining open to community input on design decisions like indexing and SRS permissions.

Big spring upgrade

All Firebird-maintained Python packages have received a major spring update. Key changes include:

- Minimum supported Python version is now 3.11
- Type hints updated to match 3.11 standards
- Tests migrated from unittest to pytest with broader coverage
- Documentation improved throughout

In addition, most packages saw various enhancements and bug fixes. Notably, a long-awaited maintenance release of the legacy FDB driver is now available.

The Firebird Project is now also established as an [organization on PyPI](#).



Toolbox: Firebird in VS Code

In recent years, Visual Studio Code has become a popular choice among developers for its flexibility, extensive plugin ecosystem, and cross-platform support. While Firebird has traditionally been underrepresented in mainstream development tools, the situation is steadily improving thanks to dedicated community efforts. This review explores two notable extensions that bring Firebird support to VS Code: **DB Explorer for Firebird Databases** and **SQLTools Firebird Driver**. We take a closer look at their features, usability, and current limitations to help Firebird developers integrate their favorite database more smoothly into their everyday workflow.

DB Explorer For Firebird Databases

As Visual Studio Code continues to evolve into a true cross-platform development environment, Firebird users have started seeing better integration options for their preferred database engine. One of the most notable tools in this area is DB Explorer for Firebird Databases, a Visual Studio Code extension that brings native database browsing and querying features directly into the editor. However, this tool comes with a story of its own.

Originally developed by Marin Vitlov, the extension introduced much-needed Firebird support into VS Code. Sadly, the project has not seen updates since its release over five years ago and appears to have been abandoned. In response, developer ninoDeme created a maintained fork, which not only preserves the original concept but also introduces new features and ensures compatibility with modern versions of Firebird and Visual Studio Code. For this review, we evaluate the forked version, which is available under the name DB Explorer for Firebird Databases Fork on the Visual Studio Marketplace.

Key Features

This extension provides next features:

- Manage multiple database connections
- List hosts, databases, tables, and fields inside the Explorer view
- Run Firebird SQL queries directly in a built-in editor
- Run predefined custom queries
- View results in a tabular format
- Realtime filtering and sorting of query results
- Export results to JSON, CSV, XLSX, or PDF
- Firebird reserved words code completion
- Table and field name code completion
- SQL mock data generator for testing and prototyping

These features make the extension not just a lightweight explorer, but a fully interactive development companion suitable for anything from debugging queries to performing lightweight data analysis or building quick prototypes.

To install the extension:

- Open the Extensions sidebar in VS Code (Ctrl+Shift+X).
- Search for "DB Explorer for Firebird Databases Fork" or visit the marketplace listing.
- Click Install.

After installation, a new “DB Explorer” section appears in the Activity Bar. From there, you can add and manage database connections through a configuration wizard or by manually editing the JSON-based connection settings.

The extension uses the `node-firebird` Node.js driver for database communication. While robust and stable, this driver depends on Firebird’s legacy network protocol and authentication model, which are no longer enabled by default in modern Firebird versions (3.0 and above). As such, you must modify your `firebird.conf` file to restore compatibility.

Here’s what needs to be done:

Firebird 3.0:

```
AuthServer = Srp, Legacy_Auth
WireCrypt = Enabled
UserManager = Legacy_UserManager
```

Firebird 4.0 and 5.0:

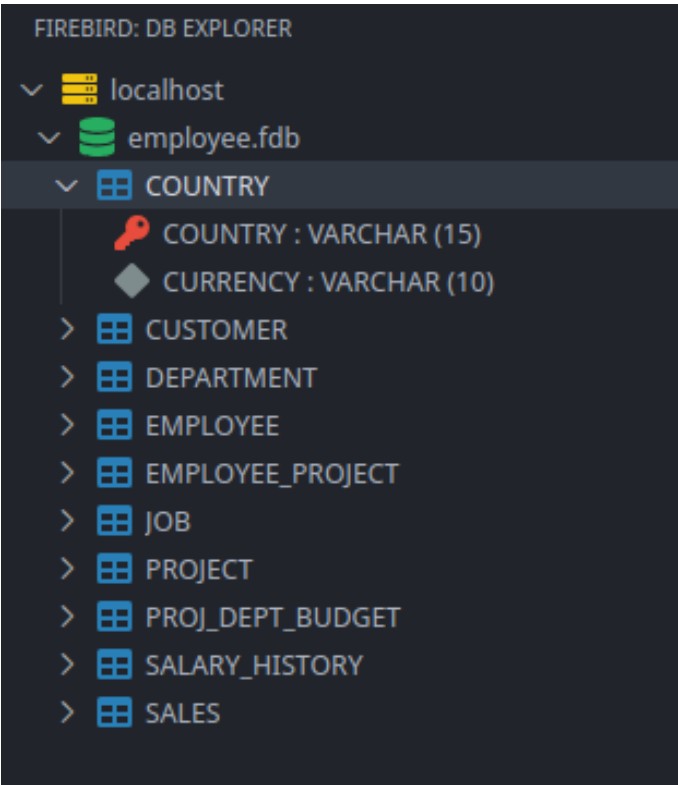
```
AuthServer = Srp256, Srp, Legacy_Auth
WireCrypt = Enabled
UserManager = Legacy_UserManager
```



Use these settings with care. While they are fine for development and local testing environments, they reduce the security hardening introduced in newer Firebird versions. In production, consider isolating these configurations to trusted machines or legacy access layers.

First thing, you need to add new connection to your Firebird database by clicking the **Add New Connection** icon in the DB Explorer title bar. You will be presented with a Connection Wizard to guide you through the process. After the process is complete, your database connection will appear inside DB Explorer View.

First thing, you need to add new connection to your Firebird database by clicking the **Add New Connection** icon in the DB Explorer title bar. You will be presented with a Connection Wizard to guide you through the process. After the process is complete, your database connection will appear inside DB Explorer View.



Most functionality is accessible through context menu on individual objects.

Host:

- Remove host

Database:

- Show database information
- Set Active
- New Query
- Remove Database

Show database information

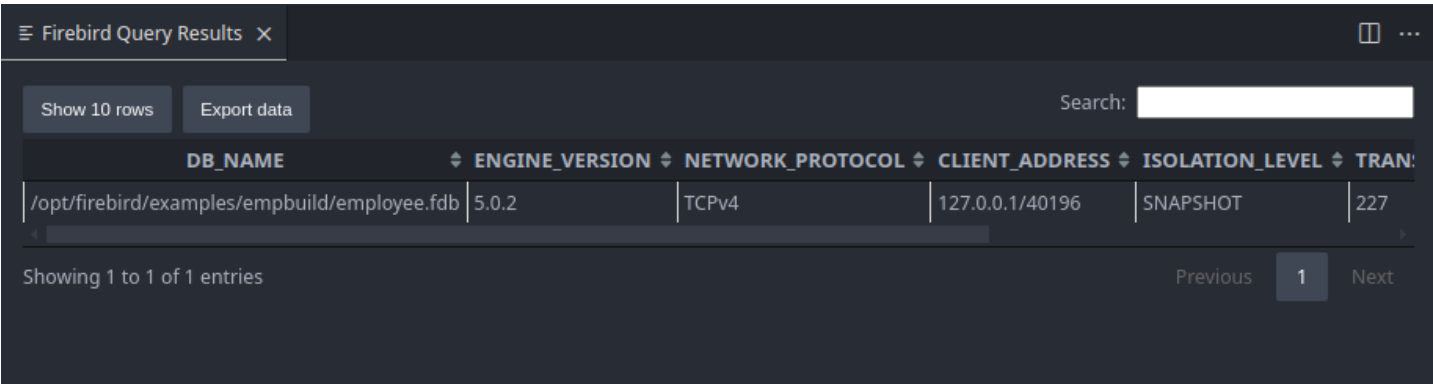
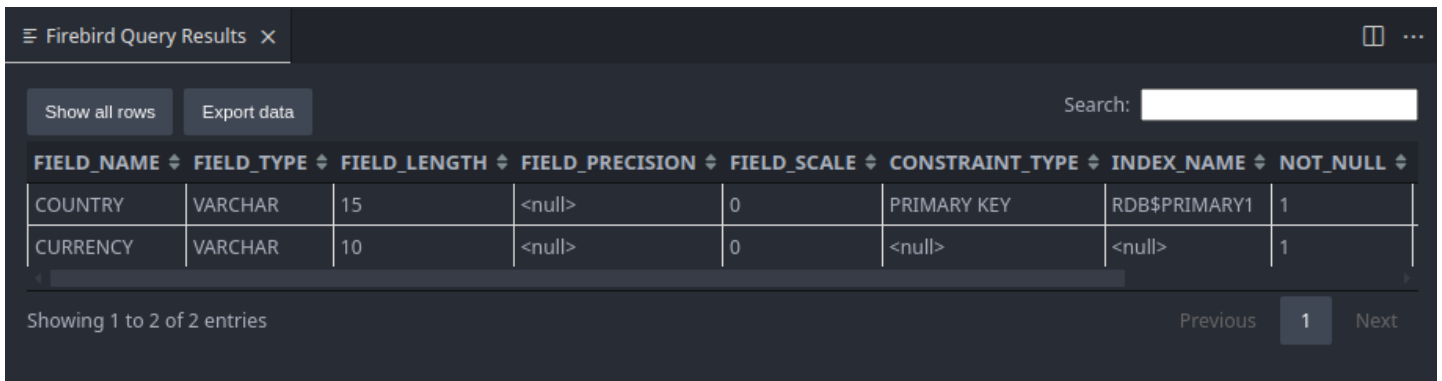


Table:

- Generate Mock Data
- Show table information
- Select all records
- Drop table

Show table information



The screenshot shows the 'Firebird Query Results' window. It has a search bar and buttons for 'Show all rows' and 'Export data'. Below is a table with 8 columns: FIELD_NAME, FIELD_TYPE, FIELD_LENGTH, FIELD_PRECISION, FIELD_SCALE, CONSTRAINT_TYPE, INDEX_NAME, and NOT_NULL. It contains two rows of data for the COUNTRY and CURRENCY tables. At the bottom, it says 'Showing 1 to 2 of 2 entries' and has 'Previous', '1', and 'Next' navigation buttons.

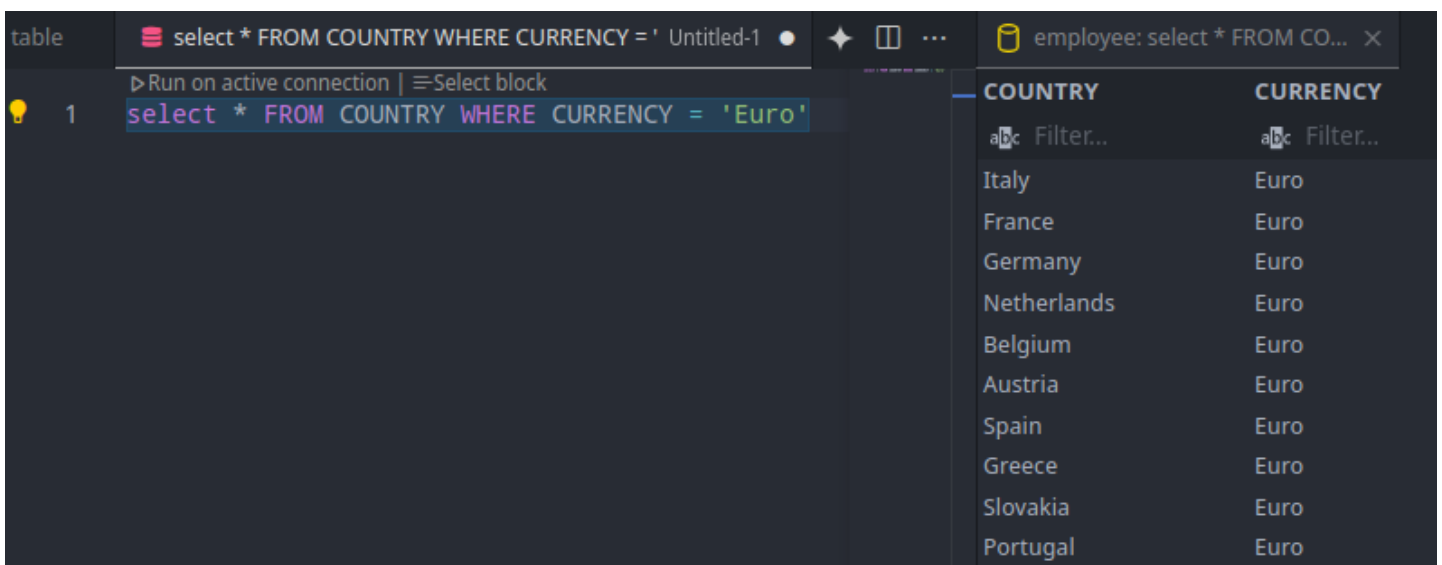
FIELD_NAME	FIELD_TYPE	FIELD_LENGTH	FIELD_PRECISION	FIELD_SCALE	CONSTRAINT_TYPE	INDEX_NAME	NOT_NULL
COUNTRY	VARCHAR	15	<null>	0	PRIMARY KEY	RDB\$PRIMARY1	1
CURRENCY	VARCHAR	10	<null>	0	<null>	<null>	1

Column:

- Select all field records

Before exploring the database content, it's necessary to set the active database. However, the **New Query** command sets the selected database active and creates new SQL document.

Query editor and Query result

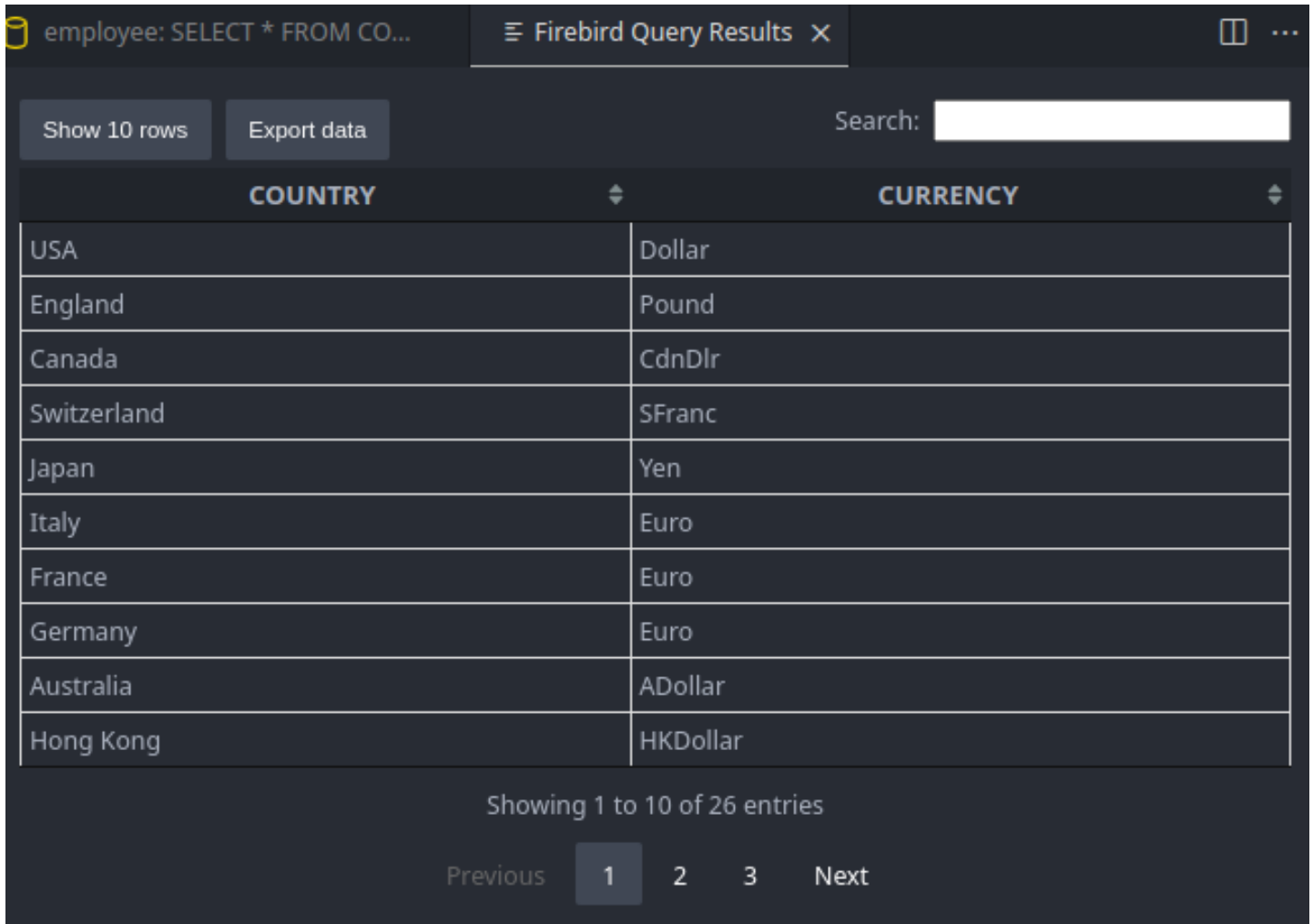


The screenshot shows a query editor with a SQL query: `select * FROM COUNTRY WHERE CURRENCY = 'Euro'`. The results window shows a table with two columns: COUNTRY and CURRENCY. The results are filtered to show only countries with the currency 'Euro'.

COUNTRY	CURRENCY
Italy	Euro
France	Euro
Germany	Euro
Netherlands	Euro
Belgium	Euro
Austria	Euro
Spain	Euro
Greece	Euro
Slovakia	Euro
Portugal	Euro

Command **Select all records** offers pagination a data export.

Result of Select all records command

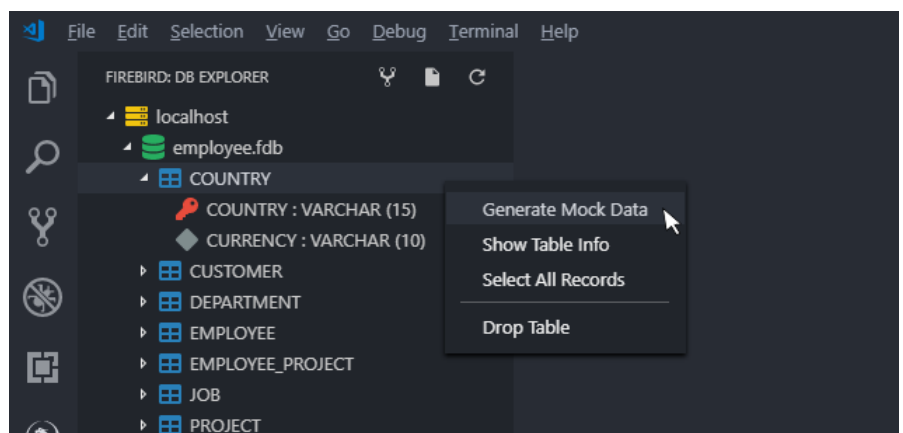


The screenshot shows the Firebird Query Results window. At the top, the query is `employee: SELECT * FROM CO...`. Below the query bar, there are buttons for "Show 10 rows" and "Export data", and a search bar. The table has two columns: "COUNTRY" and "CURRENCY". The data rows are as follows:

COUNTRY	CURRENCY
USA	Dollar
England	Pound
Canada	CdnDlr
Switzerland	SFranc
Japan	Yen
Italy	Euro
France	Euro
Germany	Euro
Australia	ADollar
Hong Kong	HKDollar

Below the table, it says "Showing 1 to 10 of 26 entries". At the bottom, there are pagination controls: "Previous", "1" (selected), "2", "3", and "Next".

The **SQL Mock Data Generator** tool allows you to generate SQL mock data that you can use to populate an empty table in your Firebird Database. It uses [Mockaroo](#) API backend, and the result is formatted as sequence of SQL INSERT statements.



SQLTools Firebird

For developers who work with multiple database systems—or simply prefer a unified interface—the SQLTools Firebird extension offers a powerful alternative to single-purpose tools. Rather than being a standalone extension, SQLTools Firebird is a driver plugin for the broader SQLTools extension, a popular database management framework for Visual Studio Code that supports a wide range of databases including PostgreSQL, MySQL, SQLite, Oracle, and more.

This design makes SQLTools Firebird especially appealing to developers working across multiple platforms, offering consistent UI, shared keyboard shortcuts, and centralized connection management within one environment.

After installation, you can define new connections through the SQLTools connection manager, available in the sidebar or via the Command Palette.



Like the **DB Explorer**, this extension uses the `node-firebird` driver under the hood. Therefore, the same changes to `firebird.conf` are required to connect successfully to Firebird 3.0, 4.0, or 5.0 (see previous section for details).

COUNTRY	CURRENCY
USA	Dollar
Australia	ADollar
Hong Kong	HKDollar
Fiji	FDollar

While **SQLTools Firebird** offers basic functionality comparable to **DB Explorer**—such as query execution, result viewing, and connection management—it also brings several distinct advantages:

- Unified interface for multiple databases: Ideal for developers switching between different backends.
- Query history and favorites: Easily recall or bookmark past SQL commands.

Conclusion

Both extensions are built with a clear goal: to support developers where they spend most of their time—inside their code editor. Rather than aiming to replace full-featured database administration tools, they focus on what matters during daily development: writing, testing, and refining the SQL queries that power your applications. For deeper database design or management tasks, more specialized tools still have their place—but for integrating Firebird into your everyday workflow, these extensions do the job well.

Resources

1. [Visual Studio Code](#)
2. [DB Explorer For Firebird Databases Fork](#)
3. [SQLTools Firebird](#)
4. [SQLTools](#)



Answers to your questions

Documentation is said to be a collection of answers to unspoken questions. If you ask a search engine, it will answer you with a link to a document that (hopefully) contains the answer. There are documents, forums and entire systems like Stack Overflow that consisting only of questions and answers. And now an army of AIs is starting to chase us to answer our questions. Questions and answers cannot be avoided, there is no hiding place.

However, amidst the sea of routine questions and responses, there lie truly captivating inquiries and answers, like hidden treasures. Our commitment is to regularly present you with a curated collection of these precious gems.

This time about:

- String equality match pitfall
- How to shred records
- How Oldest Transaction can be greater than Oldest Active?

String equality match pitfall

Dipesh asked:

When i equate the two string 'abc' = 'abc ' that also matches. So is it a bug or it is designed like this?

Ann W. Harrison answers:

The SQL standard specifies that trailing blanks in string fields are not significant in equality matches and the index uses equality matches. LIKE follows different rules. If you want to differentiate between 'abc' and 'abc ', define the field as character set octets.

Example:

```
SQL> select 'abc' = 'abc ' from rdb$database;  
=====  
<true>
```

```
SQL> select _octets 'abc' = _octets 'abc ' from rdb$database;  
=====  
<false>
```

How to shred records

anonymous asked:

Is there any way to shred deleted records so that they cannot be recovered?

Helen Borrie answers:

My question is what evidence do you have that it is possible to recover deleted records from a Firebird database. :-)

I'm not really trying to be a smartass...but usually we get the opposite question: is it really impossible to retrieve records that were deleted by mistake? (It is! well, nearly.... There are complex tools and procedures in very expert hands that may be able to identify and recover some recently-deleted data if the database has been taken off-line immediately the mistake occurred).

The main thing that makes retrieving deleted records between "difficult" and "impossible" is the fact that Firebird doesn't store table data in physical tables at all (c.f. Access, MySQL, MSSQLServer, DBISAM, etc., which do literally maintain physical files within internally controlled directory structures...find the files and you've found the tables, indexes, etc. These products often have "undelete" tools that can be used by chimpanzees.)

In Firebird and InterBase, once a delete is committed the deleted record version is marked for garbage collection. At that point those experts with their abstruse tools have a chance of getting inside the database file and sifting through the garbage in the hope of finding something. The worse the application code, the better the chances. If you have nice clean application code that doesn't leave obsolete record versions around, the normal course of events for deleted records is sudden death by decapitation...that is, they are replaced by a stub which will disappear next time you do a sweep. After that, the physical space where the record was before is just a campsite waiting for new tenants.

Of course, if you do a backup and restore, even the empty space is gone. Back versions and stubs are not backed up.

Oldest Transaction greater than Oldest Active?

Bernard Cleary asked:

I have a situation I just cannot understand and I don't think I have read about.

Oldest transaction	3822
Oldest active	3816
Oldest snapshot	3816
Next transaction	3823

The bit I don't understand is how the Oldest Transaction can be greater than the Oldest Active. This information was taken when there was no connections to a database and a manual sweep had been done.

Vlad Khorsun answers:

This is usual picture if you look at header page just after the sweep performed in exclusive mode (or if there was no transaction started during the sweep).

When any transaction starts it:

1. calculated new transaction number as follows:

1. fetched header page
2. incremented Next transaction
3. update header page with transaction numbers stored in memory variables (actually members of dbb)
4. writes header page on disk

2. calculated new OIT, OAT and OST numbers and:

1. stored OIT, OAT and OST in dbb members (dbb_oldest_transaction, dbb_oldest_active, dbb_oldest_snapshot respectively)
2. stored OIT and OST in transaction members (tra_oldest and tra_oldest_active respectively)

When sweep transaction successfully ends it assigns $OIT = tra_oldest_active - 1$ and writes this new OIT number at header page. Sweep transaction do not update OAT and OST at header page at this moment thus they leave with old values

If there are no concurrent transactions at time when our transaction starts then our transaction always calculated $OAT = OST = Next$

This is not a bug, i think. Must note that OAT and OST have sense (for engine) only if there are active transactions. In our case there are no transactions after the sweep ends. When new transaction started they will recalculate all numbers but write on disk only new Next value. If second transaction started in the same engine process they will update all numbers on disk while bumping Next value.



Planet Firebird

A regular summary of recent activities and initiatives within the Firebird database community.

Firebird at ITDevCon

At the ITDevCon Spring Edition conference held in Rome on May 23, 2025, the Firebird Foundation participated as an Open Source Sponsor. This gave us the opportunity, free of charge, to talk about the upcoming features planned for Firebird 6.

Specifically, the topics covered in Fabio Codebue's speech were TABLESPACE, SCHEMA, and the JSON datatype. The audience, made up of about forty people, immediately showed great interest in these new features, especially in SCHEMA. At the end of the session, there were many questions, including several about the implementation details of the JSON data type.

There is clearly a lot of excitement in the Italian community, which is eagerly

awaiting these new features in order to try them out firsthand.

More Firebird Certified Professionals

The Firebird Foundation's certification program for developers and DBAs is growing. In the first quarter, more candidates took the exam, and two earned the title of Certified Firebird Professional—congratulations to them!

Online exams are coming soon, along with new updates to the certification program. Learn more at [our website](#).

Mark Your Calendar: July 31st

The Firebird Project is turning 25 on **July 31st, 2025**—a major milestone for one of the most enduring open-source database engines. To mark the occasion, we're preparing something special for the entire community. Be sure to visit the **Firebird Project website** on that day for a surprise and join us in celebrating 25 years of innovation, resilience, and collaboration. Don't miss it!

Share Your News with the Community

Have something to share about Firebird? Whether you blog, build with Firebird, or have news to spread, let us know—your input helps keep the community informed and connected.

Reach us anytime at either [emberwings](#) or [foundation](#) at [firebirdsql.org](#).



Introduction to Saturnin

By Pavel Císař (IBPhoenix)

In early 2019, the Firebird Project took a decisive step toward modernizing its ecosystem with the launch of the Firebird Butler initiative. If you've read the article about Firebird Butler published in last issue, you're already familiar with the pain points that motivated Butler's creation—fragmented tooling, limited automation, and the constant need for custom workarounds in real-world infrastructure. Butler introduced a clear blueprint: define protocols, standardize service behavior, and create an open ecosystem where tooling can evolve modularly.

Saturnin is the first full implementation of the Firebird Butler architecture, offering a framework for building, deploying, and orchestrating services and microservices tailored to Firebird environments.

This article will walk you through what Saturnin is, how it realizes the Butler design principles, and why it's a foundational piece for anyone looking to automate and scale Firebird operations going forward.

Getting started with Saturnin

The best way to explain a complex system like Saturnin is through real-world use cases. Instead of starting with architecture or terminology, it's more effective to show how Saturnin supports actual workflows—beginning with what it offers administrators and operators who deploy and manage services, then shifting to the developers who build on the platform.

Saturnin is delivered through three separate packages, each with a clear role:

- `saturnin`

This is the core runtime. It provides various service containers and the Saturnin Console, a command-line interface for managing services, applications, and recipes. Installing this package sets up a standard Saturnin deployment.

- `saturnin-core`

A collection of general-purpose, open-source services provided by the Firebird Project. These include building blocks like file I/O handlers, text processors, and Protobuf encoders/decoders. Useful as-is or as reference implementations.

- `saturnin-sdk`

A toolkit for developers building new services or applications. It adds development utilities, templates, and practical examples to help speed up compliant service creation.

Installation

Since Saturnin is written in Python, you will need Python 3.11 or higher installed. For testing or production deployment, we recommend installing Saturnin using the `pipx` tool, which installs it into an isolated virtual environment, with the command:

```
> pip install pipx
> pipx install saturnin
```



If you are not familiar with the Python language environment, we recommend that you familiarize yourself with the content of the article "Using tools written in Python" from issue 2024/1.

Initialization

Saturnin uses a number of files and directories whose location in the file system corresponds to the standards for the platform on which it is installed. This basic directory placement scheme can be changed by using the `SATURNIN_HOME` environment variable, which sets the root of the other directory locations. Alternatively, you can create a "home" subdirectory in the root directory of the virtual environment in which Saturnin is isolated with command:

```
> saturnin create home
```



Because on Linux or MacOS the default location of some directories may require higher than normal access rights, we recommend that you always use the home directory setting on these platforms.

The final step is to initialize the Saturnin installation with the command:

```
> saturnin initialize
```

Saturnin console

The current version of Saturnin (0.10) is controlled using command line tools, with the Saturnin console playing a central role. It can be operated in two modes:

- **Direct (single command) mode.** The required command and parameters are entered directly on the command line, and after the command is executed, the tool is terminated.
- **Interactive console mode** activated by running the tool without additional parameters. The interactive console offers an enhanced command line with persistent command history, command and parameter completion, and interactive help.

The command set available in each mode slightly differs, as some commands (typically those required to run only once or not very often like `initialize` or `create home`) are available only in direct mode. For technical reasons, the commands that manage packages are only available in interactive mode.

For normal work, we recommend using the interactive console mode, and in the following sections, all the commands described are entered in the interactive console.



For best experience with Saturnin console and other tools, we recommend to use terminal with good support for ANSI escape sequences. On Windows platform, we recommend to use [Windows Terminal](#).

Installing services

Immediately after installation and initialization, Saturnin does not provide any Butler services to work with. To install basic set of services provided by Firebird Project, start the Saturnin Console in **interactive mode**, and use command:

```
> install package saturnin-core
```

You can now inspect your newly installed services.

```
> list services
```

Registered services

Service	Version	Description
saturnin.binary.reader	0.1.1	Binary data reader microservice
saturnin.binary.writer	0.1.1	Binary data writer microservice
saturnin.firebird.log.fromsrv	0.2.1	Firebird log from server provider microservice
saturnin.firebird.log.parser	0.2.1	Firebird log parser microservice
saturnin.firebird.trace.parser	0.1.1	Firebird trace parser microservice
saturnin.firebird.trace.session	0.1.1	Firebird trace session provider microservice
saturnin.proto.aggregator	0.2.1	Protobuf data aggregator microservice
saturnin.proto.filter	0.2.1	Protobuf data filter microservice
saturnin.proto.printer	0.2.1	Protobuf data printer microservice
saturnin.text.linefilter	0.2.1	Text line filter microservice
saturnin.text.reader	0.2.1	Text reader microservice
saturnin.text.writer	0.2.1	Text writer microservice

```
> show service saturnin.firebird.log.parser
```

Saturnin service	
UID:	a93975c3-d8c4-5e19-b898-09391cafa8d8
Name:	saturnin.firebird.log.parser
Version:	0.2.1
Vendor:	c26a6600-d579-5ec7-a9d6-5b0a8b214d3f
Classification:	firebird-log/parser
Description:	Firebird log parser microservice
Facilities:	
API:	
Distribution:	saturnin-core

Saturnin uses UUIDs based on ISO OIDs. If you prefer to see names associated with UUIDs, you need to download the specifications from the Firebird OID namespace using the `update oids` command.

```
> update oids
Downloading OID specifications starting from: https://.../root.oid ... OK
Parsing OID specifications ... OK
Updating OID registry ... OK
OID registry saved to: /home/pcisar/.local/pipx/venvs/saturnin/home/data/oids.toml
> show service saturnin.firebird.log.parser
```

Saturnin service	
UID:	a93975c3-d8c4-5e19-b898-09391cafa8d8
Name:	saturnin.firebird.log.parser
Version:	0.2.1
Vendor:	The Firebird Project
Classification:	firebird-log/parser
Description:	Firebird log parser microservice
Facilities:	
API:	
Distribution:	saturnin-core

Minimal necessary configuration

Saturnin uses several configuration files with classic `.INI` structure that could be listed with `list configs` command.

```
> list configs
```

Configuration files	
Main configuration	✓ /home/pcisar/.local/pipx/venvs/saturnin/home/config/saturnin.conf
User configuration	✓ /home/pcisar/.config/saturnin/saturnin.conf
Console theme	✓ /home/pcisar/.local/pipx/venvs/saturnin/home/config/theme.conf
Firebird configuration	✗ /home/pcisar/.local/pipx/venvs/saturnin/home/config/firebird.conf
Logging configuration	✗ /home/pcisar/.local/pipx/venvs/saturnin/home/config/logging.conf

A new Saturnin installation does not contain configuration files for the Firebird driver and logging. Saturnin does not define the Firebird driver as a dependency, as it can be used for solutions that do not work with Firebird at all, so this configuration is optional.

Logging is also optional, although recommended. However, for purpose of this article we'll skip the logging configuration and create only configuration for Firebird driver that was installed together with `saturnin-core` package using next command:

```
> create config firebird
```

The created configuration defines server named `local` with user `SYSDBA` and password `masterkey`, which is typically enough for tests accessing local Firebird instance and databases. You can adjust the configuration with `edit config firebird` command that will open it in default editor (which could be defined via `EDITOR` environment variable or `main` Saturnin configuration file).

At this point, we can start playing with the installed services.

Recipes: Orchestration Made Declarative

Recipes are Saturnin-specific configuration files with instructions for running Butler services built for Saturnin. Recipes can be created with the `create recipe` command, which creates a recipe template that typically needs to be modified further (because it only contains default values). Recipes created independently (e.g. by a solution supplier or provided by installed Saturnin application) must be installed with the `install recipe` command. Recipes have a classic `.INI` file structure, and can contain comments and interpolation references. For example:

```
; this is a comment
# this is also comment
[section]
parameter1 = value
parameter2 =
    this is
multiline value
```

```
[another.section]
parameter1 = value
; interpolation for value in the same section
parameter2 = ${parameter1}
; interpolation for value in different section
parameter3 = ${section:parameter1}
```

Since an example is worth a thousand words, let's create a simple recipe that displays the contents of a log retrieved from a Firebird server. The `saturnin-core` package provides two microservices that together handle the entire process:

- **saturnin.firebird.log.fromsrv** that fetches the log content from Firebird server and sends it to data pipe for further processing.
- **saturnin.text.writer** that receives text data from the data pipe and writes them to a file - including standard output.

To create such recipe, it's necessary to run `create recipe` command with required recipe name and list of services that should be used:

```
> create recipe test saturnin.firebird.log.fromsrv saturnin.text.writer
```

It will assemble required information into a recipe, and open it in default editor. It will mostly contain comments with available configuration options, their description and default values that must be edited to actually do anything useful. For our purpose it's necessary to define values for next options:

```
[component_1]
pipe = pipe-1
pipe_address = inproc://pipe-1
pipe_mode = bind
pipe_format = text/plain;charset=utf-8
server = local

[component_2]
pipe = pipe-1
pipe_address = inproc://pipe-1
pipe_mode = connect
pipe_format = text/plain;charset=utf-8
filename = stdout
```

Once saved, we can inspect the recipe with `show recipe` command:

```
> show recipe test
```

Recipe Metadata	
Name:	test
Type:	Bundle
Execution Mode:	Normal
Executor:	Default
Application:	Not Associated
Description:	Not provided
File Path:	/home/pcisar/.local/pipx/venvs/saturnin/home/data/recipes/test.cfg

Components in Section 'bundle'

Config Name	Service Name	Version	Description
component_1	saturnin.firebird.log.fromsrv	0.2.1	Firebird log from server provider microservice
component_2	saturnin.text.writer	0.2.1	Text writer microservice

Configuration for 'component_1' (saturnin.firebird.log.fromsrv)

Parameter	Value
agent	212657dc-2618-5f4b-a8f5-d8d42e99fe7e
logging_id	Not set (uses default)
stop_on_close	True
pipe	pipe-1
pipe_address	inproc://pipe-1
pipe_mode	2
pipe_format	text/plain;charset=utf-8
batch_size	50
ready_schedule_interval	1000
server	local
max_chars	65535

Configuration for 'component_2' (saturnin.text.writer)

Parameter	Value
agent	4e606fdf-3fa9-5d18-a714-9448a8085aab
logging_id	Not set (uses default)
stop_on_close	True
pipe	pipe-1
pipe_address	inproc://pipe-1
pipe_mode	3
pipe_format	text/plain;charset=utf-8
batch_size	5
ready_schedule_interval	1000
filename	stdout
file_format	text/plain;charset=utf-8
file_mode	3

Created or installed recipes can be run with the `run recipe-name` command. You can also get a list of recipes that can be started with the `list recipes` command.

So, we can test our newly created recipe with:

```
run test
```

that should print the content of the Firebird log from local server.

Although the range of services in the `saturnin-core` package is still small, you can already create a number of different data processing pipelines. For example, logs can be read from a file using the `saturnin.text.reader` or filtered at the text level with `saturnin.text.linefilter`. A new level of processing is introduced by the `saturnin.firebird.log.parser` service, which converts log text into a sequence of structured messages in protobuf format. It can analyze message text and extract parameters such as IP addresses, database object names, database names, and more. These can then be further processed using `saturnin.proto.filter` and `saturnin.proto.aggregator`, and finally converted into formatted text with the `saturnin.proto.printer` service.

Adding a few more services to load data from Firebird traces, monitoring tables, or gstat database structure analysis output will greatly increase the number of possibilities. And we are just getting started, and still in the data processing area. Imagine a library of dozens or hundreds of components, including services that perform specific functions such as backups, maintenance, script execution, etc. or connect Saturnin solutions to other systems.

Creating recipes, especially those that combine a large number of components, or sets of mutually cooperating recipes, is still laborious. However, once created, a recipe can be used in a wide range of contexts. We also plan to create tools (including visual ones) that will make recipe creation easier.

Applications: The Dynamic Recipes

Components have various parameters that define their behavior. A recipe sets specific values for these parameters, locking in how the component behaves. However, if some parameters could be set dynamically, the recipe would be much more flexible and usable across different contexts.

Saturnin version 0.10 introduces applications — extensions that register code to

handle recipe creation and execution. Once installed, you can list them using the `list applications` command and view details with `show application`.

Applications can register a recipe generator (`recipe_factory`) and/or a launch command (`cli_command`). Based on this, they fall into two types:

- Type `recipe`: Registers a recipe factory.
- Type `command`: Does not register a recipe factory.

If an application provides a `recipe_factory`, it can be used to create a new recipe via the `install recipe` command with the `--application` option. The method of generating a recipe is flexible — for example, it could involve a user dialog to input parameters. These applications may also provide a CLI command for the Saturnin console to run the recipe, especially if it supports additional parameters or minor adjustments before execution.

Applications that generate recipes can be used repeatedly to create multiple recipes within the same Saturnin environment.

Applications that register only a CLI command can be run directly without creating a recipe. These might generate a recipe on the fly or use a different mechanism altogether to launch Butler services.

Saturnin Architecture at a Glance

The Saturnin distribution consists of two main parts:

- Core modules for implementing Butler services in Python.
- Infrastructure for deploying and managing services on-site, including tools for managing, running, and composing these services into functional units.

Saturnin's runtime model revolves around dynamically loaded service classes and executors, which serve as container processes for these services.

Each **service** is a Python class that extends Saturnin's base service classes. Services can be packaged and distributed independently — like `saturnin-core` — while depending on the main `saturnin` package. Entry points defined in package metadata enable automatic discovery, which populates Saturnin's central service registry.

The **executor** handles loading, configuring, and initializing services based on the recipe. It ensures services start and stop cleanly and adhere to lifecycle rules. However, it does not manage message routing — that’s handled directly between services using Saturnin’s protocol stack.

Internally, the executor uses a *Service Controller* to manage services, each typically running in its own isolated thread. It communicates with services through Saturnin’s **Internal Component Control Protocol (ICCP)**, using fast in-process ZeroMQ connections.

Saturnin version 0.10 includes two service executors:

- **saturnin-service executor:** Runs a single service class.

This container can run the service either in the main thread or (by default) in a separate thread. Running in the main thread is useful for debugging and testing. It’s also better for production use when the service involves heavy computation, but does not require extensive communication with other services and clients, or external control.

- **saturnin-bundle executor:** Runs multiple services, each in its own dedicated thread.



Saturnin uses threads instead of async coroutines to keep things isolated, simple, and scalable. Each service runs in its own thread, which keeps service logic separate from the controller and avoids the mess of managing a shared async event loop. It also makes debugging easier — especially in single-thread mode with the `saturnin-service` executor, where services run plainly in the main thread. Looking ahead, Python’s upcoming support for per-interpreter GIL ([PEP 684](#) and [PEP 734](#), starting in Python 3.12) will let threaded systems like Saturnin take advantage of real parallelism through isolated subinterpreters — making this a clean, future-ready design.

Executors can also operate as background daemons using the `saturnin-daemon` utility included in the distribution.

While executors can be used directly, the preferred method is via the *Saturnin*

Console using the `run` command, which launches the appropriate executor with all necessary parameters. The console also helps to monitor and manage executors running as background daemons.

The optional `saturnin-sdk` add-on package includes an experimental `create standalone Console` command. This command generates a custom executor for a specified recipe that directly imports all required Python modules, bypassing dynamic loading. The resulting executor script can be compiled with the [Nuitka](#) Python compiler into a standalone executable. This provides improved performance when running the original recipe compared to using the default executor in a standard Saturnin setup. It also allows the application to be distributed and run independently — no need to install Python or Saturnin on the target system.

Roadmap to Version 1.0

Saturnin is ready for early adopters, but some key features are still in development for the 1.0 release.

Facilities

Saturnin is an open platform, not tied to any specific domain—including Firebird. Domain-specific features are provided via core extensions called **facilities**. Services can declare dependencies on these facilities.

The main facility available now is **firebird**, which enables integration with Firebird servers. While optional, this facility is currently hardcoded into Saturnin's core. In version 1.0, facilities will become fully modular and dynamically discoverable by the core platform.

Applications

While "Saturnin applications" are conceptually fully functional, there is still some work to be done. Especially in the area of facilitating their creation and integration into the core platform.

Services

Early Saturnin milestones focused on FBSP-based services, as shown at the 2019

Firebird Conference in Berlin. However, building and using clients for these services proved more complex than expected—mainly due to challenges with dependency management and distribution. While these issues fit naturally within Saturnin’s modular approach, they were difficult to solve within the constraints of Python 3.8 and the ecosystem’s state in 2020.

To keep momentum, we shifted focus to microservices, where these issues don’t arise. As a result, Saturnin now offers strong support for creating and using FBDP-based microservices. FBSP-based services are still supported, but working with them is more cumbersome than we’d like. There’s also no streamlined process for distributing or using FBSP clients yet.

Solving this remains a top priority. We expect to deliver a solid solution later this year.

Recipes

Currently, recipes lack support for specifying dependency versions. We also plan to enable programmatic execution of recipes—for example, triggering them directly from services.

Services in saturnin-core

The `saturnin-core` package includes 12 FBDP-based microservices. We aim to improve existing services and expand the collection. Planned additions include:

FBDP microservices:

- `gstat` analysis data source and parser
- Monitoring table data source
- Firebird server information source
- Database information source

FBSP services for Firebird-related operations:

- Backup and restore using `gbak` and `nbackup`
- Maintenance tasks via `gfix` (e.g. sweep)
- Rebuilding indices and updating index statistics
- Termination of connections and transactions
- Execution of user-defined SQL commands and scripts
- Execution of external programs or scripts

- A scheduler service to run jobs on a defined schedule

Saturnin SDK

The SDK currently offers only a minimal set of tools and examples for developers. Expanding it is our next highest priority once the core Saturnin package nears the 1.0 release.

Documentation

Saturnin's documentation is currently split across multiple sites—one for each package: [saturnin](#), [saturnin-core](#) and [saturnin-sdk](#). While not ideal, this setup is due to technical limitations and time constraints and won't change soon.

Our goal is to create a unified documentation site that brings all content together. Until then, we'll continue improving and expanding the existing documentation across the separate sources.

Conclusion

Saturnin is quickly approaching its 1.0 release, targeted for the end of this year. The current 0.10 release is ready for early adopters and testers. Your feedback is crucial—it helps us prioritize features and focus our work on solving the most urgent needs of both Firebird users and Saturnin developers.

If you're interested in the Firebird Butler concept and its realization in the Saturnin platform, now is the ideal time to get involved. Investing your time and resources today will help shape the platform and accelerate its development. Your input can make a real impact.

Resources:

- [Saturnin home page](#)
- [Firebird Butler home page](#)



...And now for something completely different

The Solstice Mystery

Years after her discovery of Turbo Mode, Renata Alvarez had carved out a unique niche for herself in the Firebird community. Though she had chosen to keep Turbo Mode a secret, the knowledge she gained during that discovery transformed her understanding of the database engine's intricacies. Her reputation as a sharp problem-solver and her ability to unlock Firebird's hidden potential made her a sought-after consultant for complex deployments.

Yet, despite her professional success, Renata's passion for Firebird history and its hidden secrets remained a personal pursuit. She often spent late evenings delving into old mailing lists, exploring forgotten branches of the codebase, and piecing together the contributions of long-retired developers. To her, Firebird wasn't just a tool—it was a tapestry of innovation and mystery, woven by countless hands over decades.

However, it was during her day job when Renata stumbled upon her next mystery.

One summer afternoon, Renata was reviewing performance logs for a client—a weather research facility in England—running simulations on a Firebird database. As she scanned the data, something unusual caught her eye: a sudden and dramatic spike in query performance, lasting exactly ten minutes, on June 21st.

The boost was extraordinary. Queries that normally took minutes were completed in seconds. There were no indications of hardware changes, manual optimizations, or environmental factors that could explain it.

Renata was intrigued. The anomaly was too specific to be random, and the behavior felt familiar—eerily similar to what she had seen in Turbo Mode.

"Could it be related?" she murmured.

Her curiosity piqued, she began comparing logs from other systems she'd worked on, spanning various regions. As she dug deeper, a curious pattern emerged: the same unexplained boost appeared either on June 21st or December 21st, depending on the system.

At first, the inconsistency puzzled her. Why would the event occur on two different dates? She stared at the calendar on the wall, unfocused, until it clicked: the boost always aligned with the local summer solstice—June 21st in the Northern Hemisphere, December 21st in the Southern. Logs from an Australian client confirmed it—the behavior mirrored what she had observed in England, just six months apart.

The discovery raised more questions than answers. Why would Firebird's performance spike during the summer solstice? Was it intentional, or some bizarre side effect?

Remembering her previous encounter with Turbo Mode, Renata revisited the Firebird codebase, focusing on the sections where Ignis had left his cryptic fingerprints. Turbo Mode had pushed hardware to dangerous extremes, and this boost felt like its controlled counterpart.

Renata began looking for time-related code, searching for anything that might tie into the phenomenon. After hours of combing through obscure modules, she stumbled upon a file she'd never seen before: `soltime.c`.

At first glance, `soltime.c` appeared to handle mundane timing functions. But

deeper examination revealed something far more complex.

In the code, Renata found a function labeled `SolAlignment()`, and the surrounding comments were unmistakably Ignis's work:

```
// Flightpaths aligned with celestial timing.  
// Duration limited to contain the flames.
```

The function tied directly into the query planner, activating fragments of Turbo Mode's optimization logic. However, instead of running indefinitely, the Solstice Code constrained its activation to a precise ten-minute window. It seemed designed to harness Turbo Mode's power safely, taming its risks through careful limits tied to the summer solstice—a deliberate choice whose true purpose only Ignis might have understood.

With the next summer solstice approaching in the Northern Hemisphere, Renata prepared to test her theory. She scheduled a resource-intensive workload to run during the predicted activation window on June 21st, configuring her server to log every detail of the system's performance.

On June 21st, exactly five minutes before local solar noon, the Firebird's behavior changed. The system came alive, processing queries with breathtaking speed. For ten minutes, the database performed at levels she had only seen during her experiments with Turbo Mode—but without any of the accompanying risks.

Renata sat at her desk, staring at the logs from her solstice test. The Solstice Code was brilliant: a fleeting moment of safe, controlled power tied to a specific time of year. But as she considered sharing her findings with the Firebird community, doubts began to creep in.

Turbo Mode haunted her thoughts. If the Solstice Code was made public, it wouldn't take long for someone to connect the dots to Turbo Mode. Worse, the community might try to manipulate the Solstice Code to extend its duration, inadvertently unleashing the catastrophic effects of Turbo Mode.

Renata needed a plan—one that would minimize the risks.

Her thoughts turned to Phoenix42, the mysterious figure who had once guided her through the dangers of Turbo Mode. Renata sent him a carefully worded message,

outlining her discovery and her concerns about revealing the Solstice Code.

Two days later, she received a reply:

“The Solstice Code? Ignis never mentioned it. But this... it sounds like him. A hidden flame to prove that Turbo Mode could work, if only for a moment. A gift—and a lesson.”

The revelation surprised Renata. Even Phoenix42, one of Ignis’s closest collaborators, hadn’t known about the Solstice Code. Ignis had worked alone to create this Easter egg, a secret demonstration of Turbo Mode’s potential when aligned with the right conditions.

After hours of discussion, Renata and Phoenix42 agreed to hold off on announcing the Solstice Code until just before December 21st—the next upcoming summer solstice. That timing would give the community a chance to prepare for testing it properly. After the event, Phoenix42 would publish a technical explanation, aiming to satisfy curiosity while discouraging the inevitable attempts to stretch or manipulate the feature beyond its safe limits.

When December approached, Renata published her findings in a carefully crafted post, describing the Solstice Code as an intentional Easter egg left behind by Ignis. She detailed its brief but powerful effects during the solstice and provided instructions for scheduling workloads to take advantage of the boost.

The community’s response was immediate. Developers were eager to verify the claim, and as December 21st arrived, excitement reached a fever pitch. On the solstice, servers across the globe sprang to life for ten minutes of unparalleled performance—each at their local solar noon.

A week later, Phoenix42 published his explanation. In a technical deep dive, he revealed how the Solstice Code leveraged fragments of Turbo Mode’s architecture while mitigating its dangers through strict timing and alignment with unknown external forces. He emphasized the risks of tampering with the feature, warning the community that attempts to extend its activation could destabilize systems or worse.

Despite the warnings, Firebird developers couldn’t resist experimenting. Over the following months, teams tried modifying the Solstice Code to activate at other

times or more frequently.

The results were always the same: the feature worked safely only at the exact moment of the summer solstice, and only for ten minutes. Attempts to force activation at other times caused performance anomalies, corrupted data, or outright crashes.

Speculation ran wild. Some believed the feature relied on geomagnetic stability unique to the solstice. Others thought Ignis had tied the timing to solar energy cycles or deliberately chosen an arbitrary constraint to avoid prolonged use.

The truth remained elusive.

Months after the solstice, Renata reflected on the impact of her discovery. The Solstice Code had brought the Firebird community together, inspiring awe and curiosity. But it also left them with more questions than answers.

Why did the code work only at the solstice? Was it truly tied to natural forces, or was it just another of Ignis's brilliant puzzles?

Phoenix42 speculated in his final message to Renata:

"Ignis had a way of encoding meaning into his work. Maybe the Solstice Code was his way of saying that even brilliance must have its limits. A flame burns brightest when it knows when to fade."

Renata smiled as she read the message. The Solstice Code was a triumph, but it was also a reminder of the mysteries still buried in Firebird's depths. Ignis had vanished years ago, leaving no trace but his work—and that work continued to captivate and confound.

Somewhere in the Firebird code, other secrets waited, each one a piece of a larger puzzle crafted by a mind far ahead of its time.

For the Firebird never truly rests, and neither do the flames it leaves behind.



The official quarterly magazine of the Firebird Project

Do you develop with Firebird?

Are you using Firebird as a database backend for your applications? Share your experience and help shape its future! Your insights on how you develop with Firebird, the tools you rely on, and your wishlist for improvements will directly impact its development. The survey takes just 5-10 minutes—join us in building a better Firebird!

[Take the Developer Experience survey](#)

Do you manage Firebird deployment?

Your insights as a Firebird administrator are invaluable! By taking just a few minutes to complete this survey, you'll help shape the future of Firebird, identify key challenges, and improve the tools and features you use every day.

[Participate in the Admin survey](#)

We value your opinion! Help us improve **EmberWings** magazine by sharing your thoughts and feedback. Our quick questionnaire will only take a few minutes, and your responses will guide us in making future issues more relevant, engaging, and valuable to the Firebird community.

[Help us improve EmberWings!](#)