EmberWings

2025/4

Firebird

December 2025

www.firebirdsql.org

emberwings@firebirdsql.org

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.

In This Issue:

The Fire That Refused to Fade

Dear Readers,

December always asks us to look back. This year, it asks a little more. Firebird turned twenty-five, and the milestone has been present in everything the community has written, debated, released, patched and dreamed since January. What began as a small, determined fork has become a database that outpaced every prediction made about it. This issue marks the final chapter in our anniversary year, and we wanted to shape it with both gratitude and clarity.

Our lead article traces Firebird from its early bold steps in 2000 through the battles, breakthroughs and quiet victories that shaped the database we rely on today. It follows the people who carried it forward when the road was uncertain, the features that defined each era and the turning points that might have gone unnoticed at the time but changed our direction in the long run. Twenty-five years is a long arc. Seeing it written out reminds us how focused work and stubborn creativity brought Firebird through every challenge.

This issue also brings our regular sections, each with its own weight. And then there is our story. Nine months ago, we started a fictional adventure about hidden features, cryptic commits and a legacy buried inside Firebird's ecosystem. It was playful, intricate and strangely fitting for a year that asked us to explore where Firebird came from. With this issue, the story reaches its conclusion. Thank you to everyone who followed it, analyzed it, or tried to guess its twists.

As we close the anniversary year, one truth stands above the rest. Firebird reached twenty-five because people cared enough to keep building, keep fixing, keep writing, keep teaching and keep supporting. The project grew through trust, shared responsibility and a stubborn respect for clean engineering. Whatever the next decade brings, these principles will carry us forward.

Enjoy the issue, enjoy the finale and thank you for being part of the journey.

Warm regards,
The EmberWings Team

The apprentice sat with the Master of Firebird beneath a quiet pavilion. A single wooden table stood between them. On it rested a bound scroll labeled only `The Work`.

The apprentice asked, “Teacher, what is it like to give twenty five years to Firebird?”

The Master unrolled a small part of the scroll. It showed a few clear lines written long ago. “Read.”

The apprentice read them. They were simple. Almost obvious.

The Master rolled the scroll forward. The writing grew more refined. Patterns emerged. Earlier mistakes were corrected by later clarity.

The apprentice said, “It changes.”

The Master shook his head. “Look again.”

The apprentice read from the start, then from the middle, then from near the end. He frowned. “The writing changes, but the hand is the same.”

“Yes,” said the Master. “This is what twenty five years truly are. You think you are building something outside yourself, but most of the building happens here.”

He touched his own chest.

The apprentice asked, “Where is the passion in this? It feels so boring.”

The Master took the scroll’s end and unrolled a single blank section. “Passion is not loud. It is the space you return to. Day after day. The blank part. The part that asks who you are becoming as you write.”

The apprentice ran his fingers across the empty parchment. It felt alive with possibility.

The Master said, “In the beginning, I wrote to prove something. Later, I wrote to fix something. Then I wrote to understand. Now I write because the act shapes me. That is passion after its noise has faded.”

The apprentice asked, “And learning?”

The Master placed the apprentice’s hand on the scroll. “Learning is when the same line teaches you something new each year. Not because the line changed, but because you did.”

The wind stirred the pavilion. The scroll fluttered like a quiet wing.

The apprentice whispered, “So Firebird grows as I grow.”

The Master smiled. “Only when you let it.”

The apprentice bowed. “Then I will keep writing.”

The Master nodded. “Do that, and one day you will find you have given many years. And none of them were wasted.”

25 Years of Firebird

By Pavel Císař (IBPhoenix)

The June issue featured "Rising of the Phoenix", the story of how Firebird took shape from the ashes of InterBase 6 in July 2000. This follow-up looks at the twenty-five years that came after. Some stretches were calm, others hectic and demanding. A few were so tough that Firebird's future was hanging by a thread. Even so, the shared effort and steady commitment of everyone who believed in the project kept it alive and moving forward. Their work is the reason Firebird is still going strong today. This article is a modest attempt to honor what they achieved.

2000: The Beginning

The Firebird project began to take shape on SourceForge in early August 2000 and quickly drew more than thirty people from around the world. In those first months, the focus was on forming a real team, setting up workflows, and getting familiar with the source code. We needed to understand what we had

inherited, what Borland had withheld, and how to produce the first binary distributions of Firebird.

The biggest obstacle was the lack of documentation, along with the old InterBase testing system (TCS), which came with only a small set of tests. Help from a few InterBase engineers at Borland, whether former or quietly supportive, made a real difference. The newly launched IBPhoenix website also played a key role. Paul Beach, Ann W. Harrison, and others began publishing articles, tutorials, documentation, and tools that opened up InterBase's inner workings. And as a final bonus, Jim Starkey, the original creator of InterBase, began participating in the Firebird discussion groups. His sharp, insightful posts quickly earned him a memorable nickname: "the Big Bad Wolf who lives in the Firebird-Architect mailing list."

Politically, Firebird was in a difficult spot. Borland ultimately chose to continue developing InterBase as a commercial product, which still exists today under Embarcadero. This left Firebird competing directly with the company it had just split from. Although there was some communication with the InterBase team led by Charlie Caro, collaboration never took shape. After everything that had happened, trust in Borland was low, and the decision was made to move forward independently.

Another complication was that the InterBase 6 code Firebird inherited was not a finished product. It was a preliminary build with a long list of flaws that had to be fixed. This is why the first official Firebird release, published in December, carried the version number 0.9.

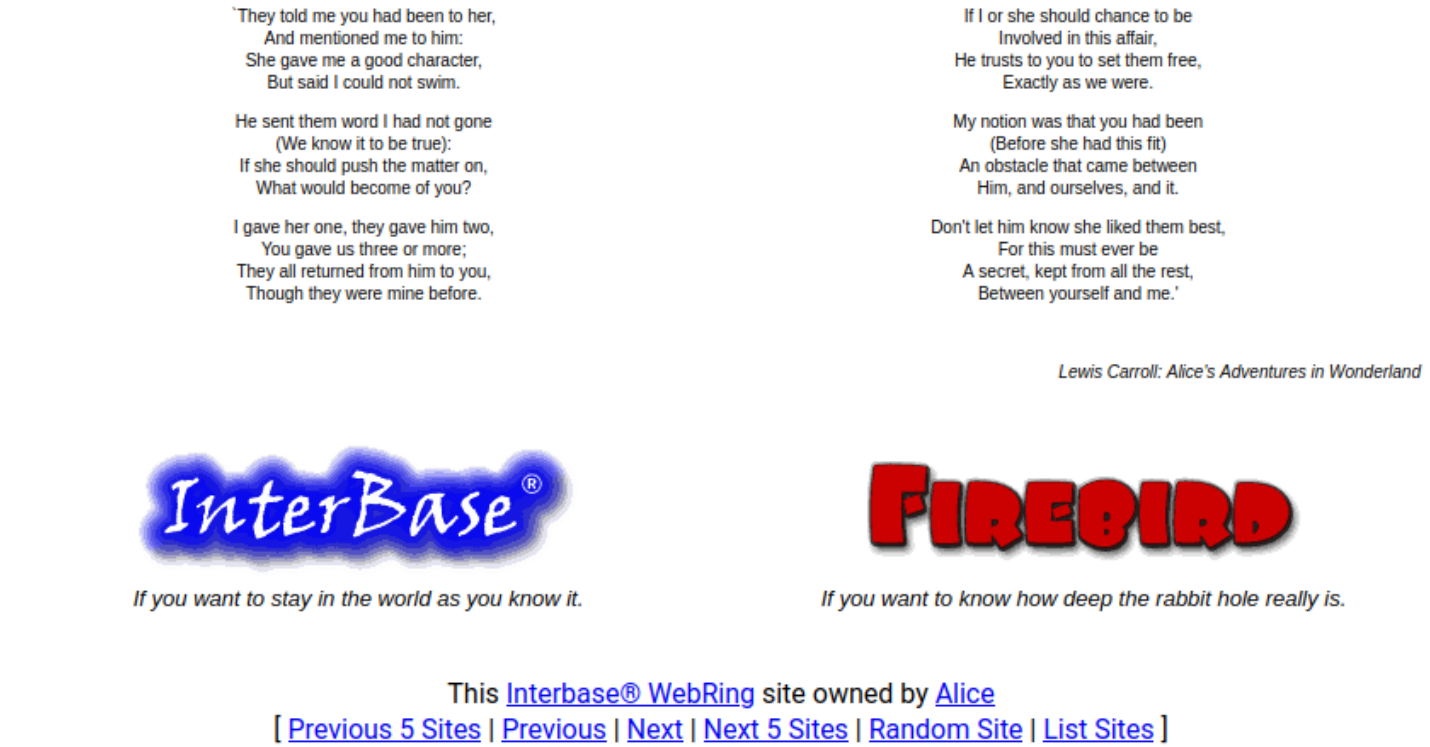
December also brought an unusual discovery that would put Firebird in the global spotlight the following year. While working through the code, Frank Schlottmann-Goedde found a hard coded password and a strange internal UDF. These security breaches had been sitting in the code since 1994. Inprise was alerted discreetly.

Global Context

Beyond InterBase, Firebird's biggest competitors at the time were Oracle, DB2, and MySQL, which was rising fast thanks to the popularity of the LAMP stack. PostgreSQL, although respected in open-source and academic circles, was still rarely chosen for large commercial projects. It was maintained entirely by a global

volunteer core team, with only limited early commercial involvement from Red Hat and the short-lived Great Bridge. PostgreSQL's real turning point arrived a year later with the release of version 7.1.

The main page of very first Firebird Project website
(From my personal archive, as Wayback machine does not have InterBase name image)
(For the record, Matrix memes were everywhere in 2000)



2001: The road to Firebird 1.0

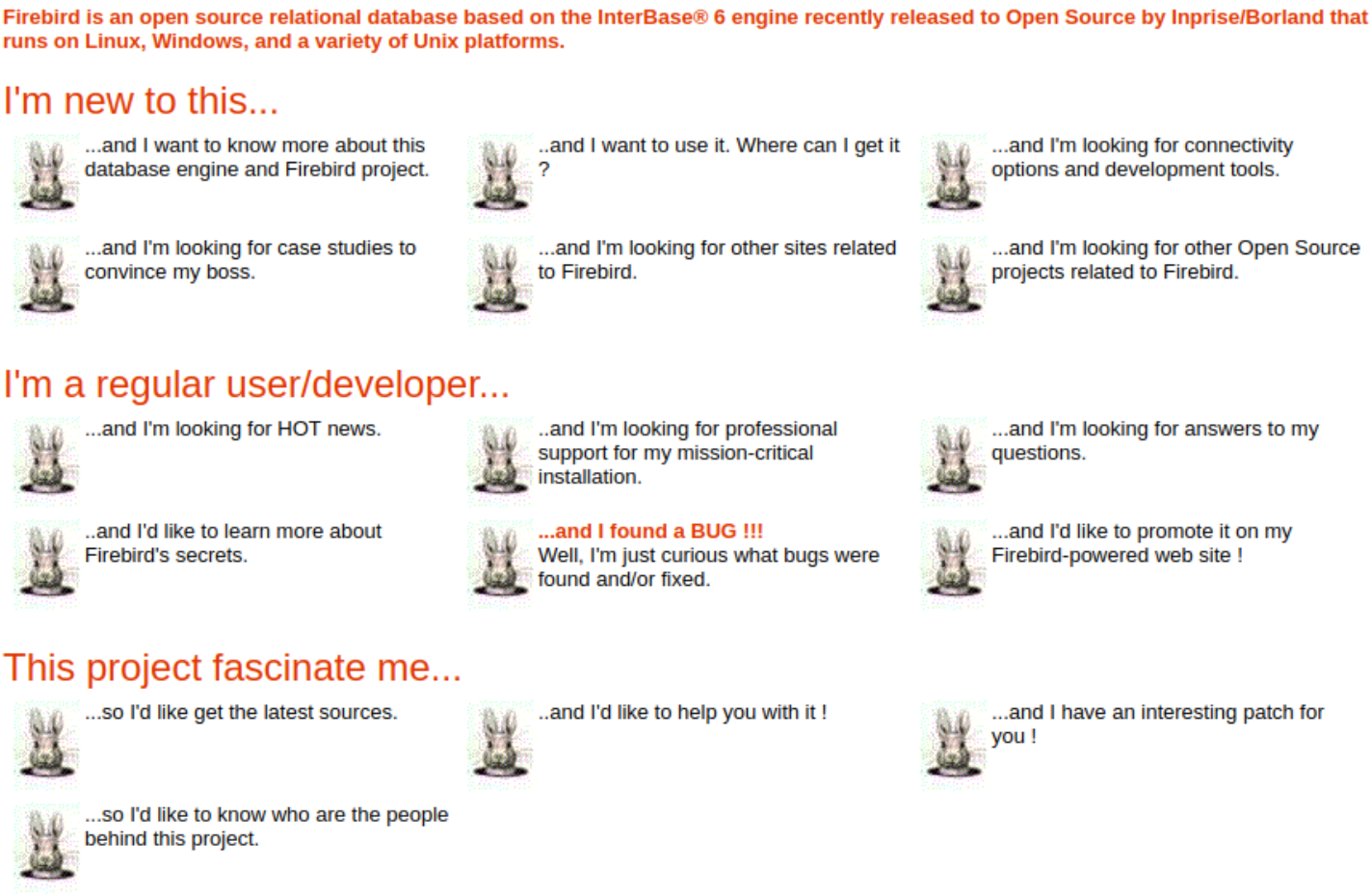
In January, Inprise returned to the Borland name, and CERT issued an advisory for all users of InterBase 4 and newer. The issue stemmed from a back door that Borland's InterBase engineering group had intentionally added around 1992. It allowed direct access to the security database (`isc4.gdb`) in version 4, and the same account and password continued to grant full access in all later versions. InterBase 4, 5, and 6 also included a built-in function that could crash the server or delete a database file, depending on the parameters passed. Borland's QA group had requested this function in 1994 to simplify testing and insisted it remain in production builds.

Jim Starkey, the original InterBase architect, developed a patch that corrected the issue for all affected InterBase versions. IBPhoenix made this fix available at no cost. Firebird 0.9-4 followed soon after, addressing these vulnerabilities along with several other potential security problems.

The backdoor story kept the IT press busy for weeks and sparked loud, spirited debate on Slashdot. It was the first time Firebird's name reached a broad audience, and the attention gave the project a welcome boost in both visibility and morale.

Throughout the year, the Firebird project focused on fixing bugs, adding small improvements, and slowly getting ready for its first major release. Support for new platforms also expanded, making Firebird available on Linux, Windows, Solaris, MacOS, and FreeBSD. In November, the project published the first Release Candidate of Firebird 1.0.

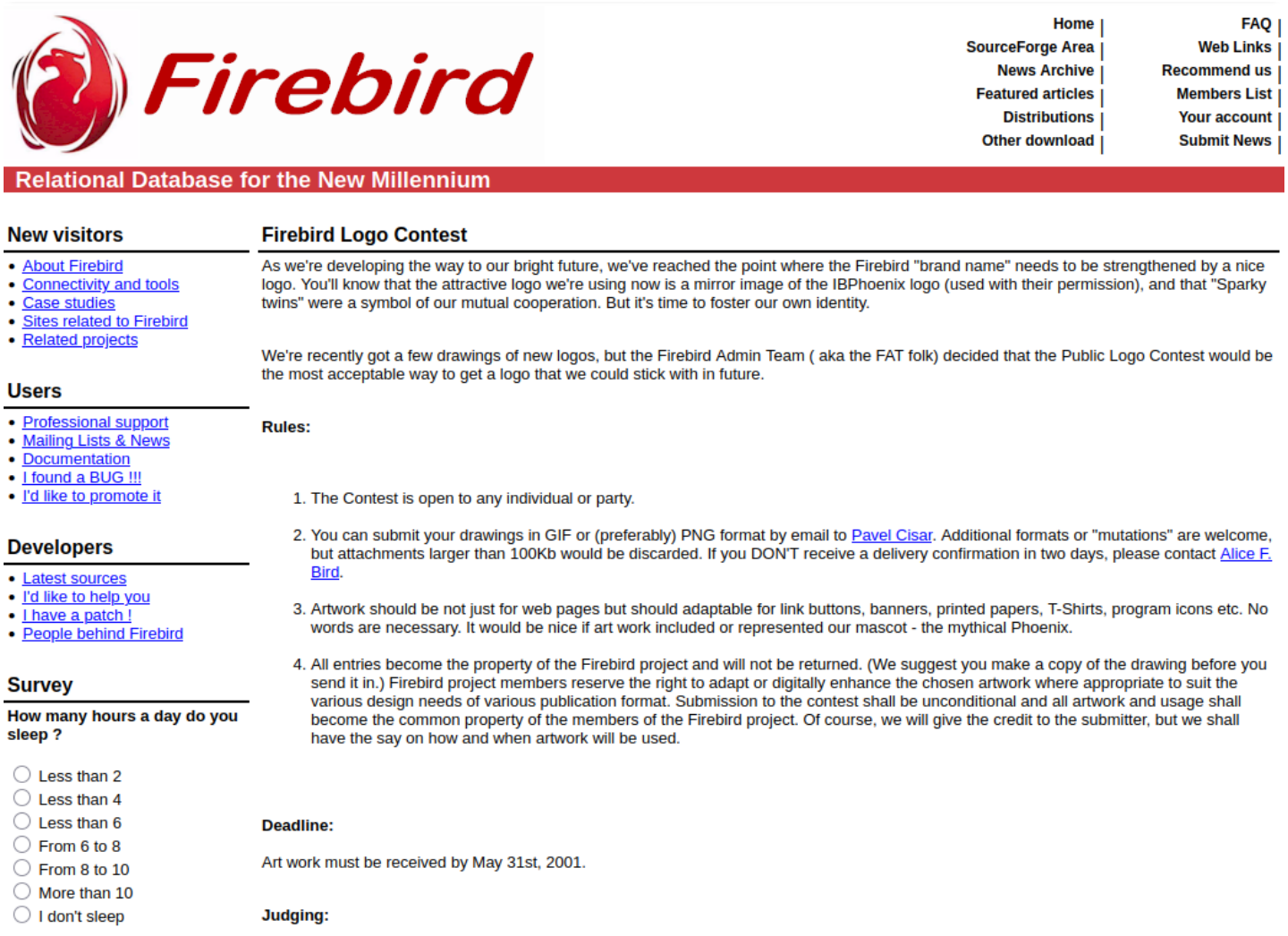
Firebird Project website when you take the Red Pill on the main page (From my personal archive, as Wayback machine does not have the rabbit image)



Other notable milestones

In March, the Firebird project relaunched its website using `phpNuke` to support dynamic, interactive content. Soon after, a public logo contest was started, as the project had reached a point where the brand needed a distinct visual identity. Until then, the project had been using a mirrored version of the IBPhoenix logo, with permission. These “Sparky twins” had symbolized close cooperation, but it was time for Firebird to stand on its own. The contest was won by Markus Soell, and Firebird still uses his design today.

The main page of Firebird Project website
phpNuke, spring 2001, with IBPhoenix mirror logo



In August 2000, IBPhoenix launched the InterBase 6.0 ODBC appeal to buy and open the ODBC code Jim Starkey had written for Borland. Firebird users contributed the required amount, and the driver was released under the Initial Developer’s Public License on 31 July, the first anniversary of the Firebird project.

Bad omens

So far it had been a successful year, but no one expected the end of it to bring two events that subsequently almost killed the young Firebird project before it could truly spread its wings.

First, in November, Aleksey Karyakin and Oleg Ivanov created a fork of Firebird called Yaffil. Then, in December, Borland announced InterBase 6.5. It was just a minor enhancements over version 6 (and some taken straight from Firebird), but it was clear sign that Borland is serious about competition with Firebird.

2002: The year of the struggle

It all started well. In March, Firebird 1.0 for Windows, Linux, and MacOS was released, soon followed by versions for Solaris, FreeBSD, and HP-UX. The website went through another redesign. The phpNuke system, hacked in December 2001, was gradually replaced by a custom PHP site built from scratch. Along with new frontend content that absorbed material from the now-dissolved IBDI, it introduced “Rabbit Holes” for project members and teams. In April, a beta version of a pure Java JCA-JDBC driver was released.

While everything looked stable from the outside, underneath we were fighting to survive. After the Firebird 1.0 release, the biggest question was what to do next. Since InterBase was written in C, the team decided the first step toward the future should be a full conversion to C++. Mike Nordell had started this work back in 2000, but most of the effort took place after 1.0. The conversion touched every corner of the engine.

Developers also followed their own interests, so work spread across many areas: optimizer updates, PSQL and DSQL improvements, new character sets, better tools and services, build workflow refinements, and more. Because the changes were mostly incremental, the next release was tagged as 1.5. Looking back, it may not have been the best choice, since it suggested slow progress and an unclear roadmap. With so much happening at once, it became difficult to keep track of everything, and the focus of post-1.0 development began to slip.

Another priority was securing a stable base for the project. In February, a steering committee was formed to explore the creation of a non-profit organisation that

could raise funds to support the volunteer developers. It took most of the year, but in November the Foundation was registered in New South Wales, Australia, and its first AGM was held in December.

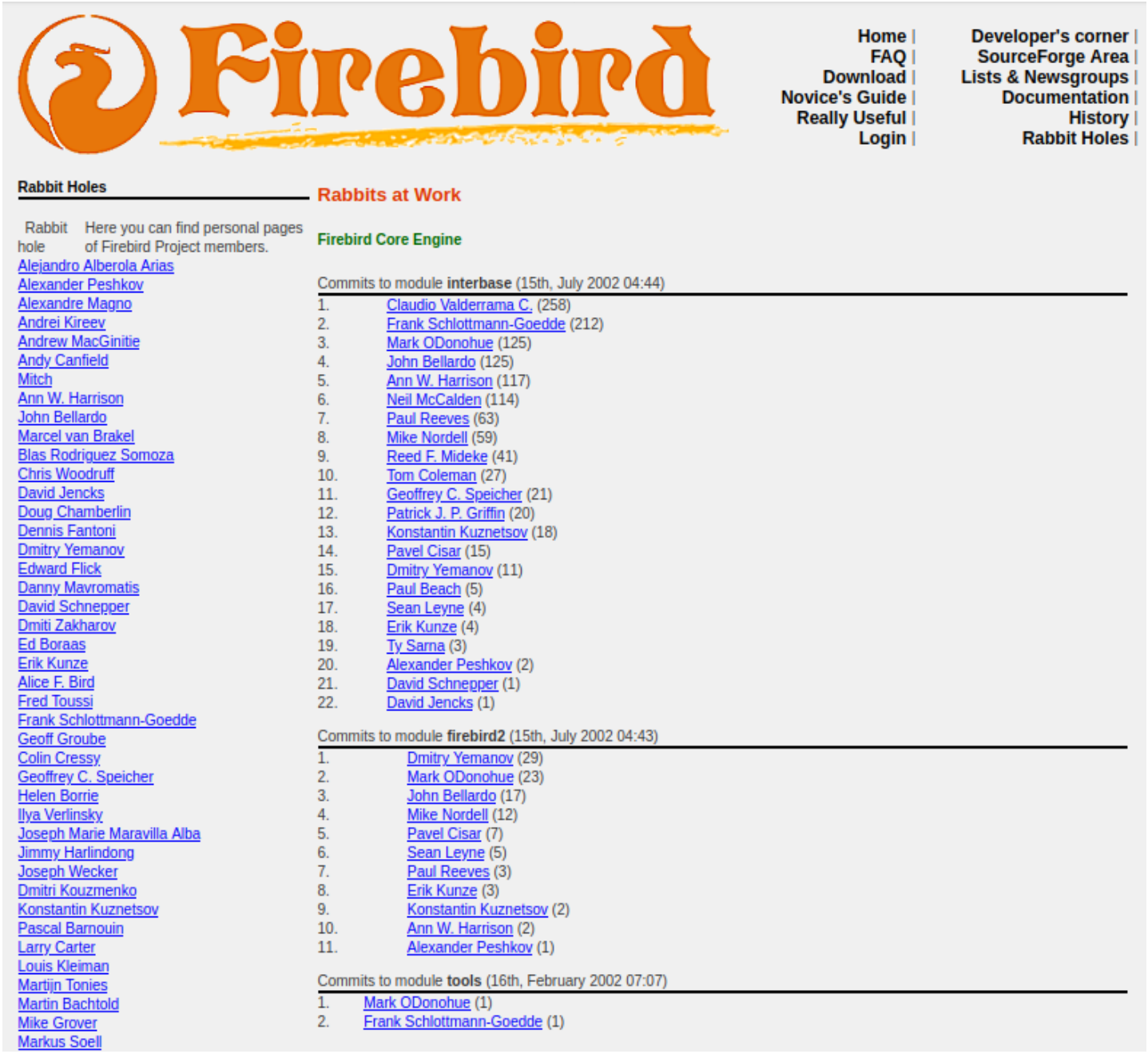
Then, in July, Borland suddenly released InterBase 7 with SMP support. It was the first serious blow to Firebird.

The second came from an unexpected direction. The Yaffil fork, although focused strictly on Windows, began delivering improvements that outpaced Firebird 1.0. It gained early traction in Russia, and over time its reputation spread globally.

Firebird was beginning to fall behind its closest relatives.

The main page of Firebird Project website
custom php site, September 2002

Rabbit Holes at Firebird Project website, September 2002



Firebird

Home | FAQ | Download | Novice's Guide | Really Useful | Login |
Developer's corner | SourceForge Area | Lists & Newsgroups | Documentation | History | Rabbit Holes |

Rabbit Holes

Rabbit hole — Here you can find personal pages of Firebird Project members.

Alejandro Alberola Arias
Alexander Peshkov
Alexandre Magno
Andrei Kireev
Andrew MacGinitie
Andy Canfield
Mitch
Ann W. Harrison
John Bellardo
Marcel van Brakel
Blas Rodriguez Somoza
Chris Woodruff
David Jencks
Doug Chamberlin
Dennis Fantoni
Dmitry Yemanov
Edward Flick
Danny Mavromatis
David Schnepper
Dmiti Zakharov
Ed Boraas
Erik Kunze
Alice F. Bird
Fred Toussi
Frank Schlottmann-Goedde
Geoff Groube
Colin Cressy
Geoffrey C. Speicher
Helen Borrie
Ilya Verlinsky
Joseph Marie Maravilla Alba
Jimmy Harlindong
Joseph Wecker
Dmitri Kouzmenko
Konstantin Kuznetsov
Pascal Barnouin
Larry Carter
Louis Kleiman
Martijn Tonies
Martin Bachtold
Mike Grover
Markus Soell

Rabbits at Work

Firebird Core Engine

Commits to module **interbase** (15th, July 2002 04:44)

1. Claudio Valderrama C. (258)
2. Frank Schlottmann-Goedde (212)
3. Mark ODonohue (125)
4. John Bellardo (125)
5. Ann W. Harrison (117)
6. Neil McCalden (114)
7. Paul Reeves (63)
8. Mike Nordell (59)
9. Reed F. Mideke (41)
10. Tom Coleman (27)
11. Geoffrey C. Speicher (21)
12. Patrick J. P. Griffin (20)
13. Konstantin Kuznetsov (18)
14. Pavel Cisar (15)
15. Dmitry Yemanov (11)
16. Paul Beach (5)
17. Sean Leyne (4)
18. Erik Kunze (4)
19. Ty Sarna (3)
20. Alexander Peshkov (2)
21. David Schnepper (1)
22. David Jencks (1)

Commits to module **firebird2** (15th, July 2002 04:43)

1. Dmitry Yemanov (29)
2. Mark ODonohue (23)
3. John Bellardo (17)
4. Mike Nordell (12)
5. Pavel Cisar (7)
6. Sean Leyne (5)
7. Paul Reeves (3)
8. Erik Kunze (3)
9. Konstantin Kuznetsov (2)
10. Ann W. Harrison (2)
11. Alexander Peshkov (1)

Commits to module **tools** (16th, February 2002 07:07)

1. Mark ODonohue (1)
2. Frank Schlottmann-Goedde (1)

2003: Reverting the tide

By the end of the previous year it was clear that Firebird 1.5, built on a cleaner and fully converted C++ codebase with many new features and fixes, had to reach users as soon as possible. A Beta version was released at the end of January.

On 6 February, HK-Software announced it would host the first European Firebird Conference, with sessions in English and German, on 19–20 May in Fulda, Germany. It was a rare chance for the project to connect directly with its user community. To help build momentum for the event, the project published the first

Release Candidate of Firebird 1.5 on 14 April.

That same day, the Mozilla project announced it would rename its emerging web browser from “Phoenix” to “Firebird.” The move was meant to avoid a trademark conflict with Phoenix Technologies, but Mozilla’s growing popularity threatened to overwhelm the identity of the Firebird database server. Firebird project raised strong objections, warning about user confusion, search engine dilution, and trademark damage. What began as a community argument quickly escalated into legal threats, mass email campaigns, and attempts at mediation. In early 2004, Mozilla finally adopted a new name: “Firefox.”

The clash was a true “David vs. Goliath” moment. Stressful as it was, it gave Firebird unprecedented public visibility. Alongside the first Firebird conference (that was a great success), it likely helped prevent the project from slipping into decline.

However, the hardest battle was still ahead. Although everyone expected Firebird 1.5 to ship during the summer, the new version turned out to be full of issues. With such massive changes to the code and only rudimentary QA tools, this was almost inevitable. Developers worked tirelessly to fix every reported bug and problem, but nine release candidates were needed before the quality was considered acceptable for a final release. Even then, despite all the effort, Firebird 1.5 did not make it out that year.

Meanwhile, work on the next major version, Firebird 2.0, began in August. The key goals were:

- Stronger scalability and performance (SMP support, improved caching, better optimizer).
- New SQL features (built-in functions, datatypes, statements, metadata object types).
- Better alignment with the latest SQL specification.
- An optimized network protocol.
- Proper, modernized security.
- Incremental backup, native monitoring features, engine plugins.
- And more…

The end of the year brought more good news. First, the principals of the Yaffil project agreed to merge it back into Firebird. With the Yaffil code, Firebird gained a set of valuable enhancements, optimisations, architectural improvements, and new character sets and collations.

The second piece of good news was the return of Jim Starkey. He was hired by IBPhoenix to produce a 64-bit, SMP-friendly reference port for Firebird, which he called “Project Vulcan.” The contract was funded by SAS Institute, who needed a thread-safe, embeddable, 64-bit version of Firebird capable of tripling performance on a four-processor system, or at least pushing the hardware to its limits. Because Vulcan was developed under the IDPL, it was expected to merge into the Firebird 2.0 codebase and deliver the long-awaited SMP support. The formal target platform was 64-bit Solaris SPARC, though Jim also planned AMD64/Opteron builds for Windows XP and Linux along the way.

With all these changes on the horizon, and after the long struggle with the 1.5 release, it was clear that the project needed a much stronger QA process. Without it, delivering future versions on time and at a stable level of quality would be extremely difficult.

Other notable milestones:

- JayBird 1.0 - the pure Java JCA-JDBC driver for Firebird 1.0 was released on 28th April.
- Firebird .NET Data Provider 1.0 was released om 25th May.
- Firebird 1.0 was included into Red Hat and Mandriva (via contribs)

Mozilla Takes Firebird’s Name for its Browser

From Firebird website, 14th April 2003

The Mozilla subproject formally known as "Phoenix" has decided to rename its lightweight browser product to "Firebird". The Firebird development team, and the larger Firebird community, strongly oppose this change.

We of the Firebird project are proud of our product and devoted to our branding. Our marks are not there for the taking and our advice is that the law is on our side: we have nearly three years of widespread international use of our mark.

Legal issues notwithstanding, we take this as a slap in the face to the entire open source community. The easy route will be for Mozilla to quickly retract what it has done and pay the open source community the courtesy of an apology. The hard route, which we sincerely hope to avoid, will be Mozilla proceeding with this, losing its Open Source cred and goodwill and taking the risk of making its product untouchable by the distributors.

We are dismayed that Firebird was not contacted before Mozilla’s decision was finalized. This breach of principle has occurred in the heartland of open source, where we are all supposed to

be above such things. The attitude adopted by Mozilla's vocal proponents of the change, in essence "if they don't like it they can sue", is contrary to the generally accepted core values of the open source community. It reflects poorly on a community that voices strong opinions when corporate entities employ similar tactics. If Open Source is to win, we can do without brother cynically stealing from brother.

Please make your feelings on this issue known by emailing and posting to the Firebird-general list.

Firebird thanks you for your support.

The Firebird Admins:
John Bellardo, *Helen Borrie*, *Pavel Cisar*, *Ann Harrison*, *David Jencks*, *Sean Leyne*, *Mark O'Donohue*, *Dmitry Yemanov*

Protecting the Firebird Brand: Update

From Firebird website, 29th April 2003

After nearly two weeks of effort to get our objections heard via the Web and private email, we are beginning to see some action by the Mozilla leaders to unroll the re-branding of its lightweight browser as "Firebird™". When this issue first blew up, we took legal advice. On Friday (April 25), we (Firebird Admins), the FirebirdSQL Foundation committee and the IBPhoenix principals wrote a formal letter to the Mozilla admins, stating the legal position and asking them to contact us. We look forward to their response.

However, over the weekend and today, we note:

- a branding policy statement that should leave nobody with an excuse to err, and
- some observable activity to change or disable web pages containing trademark infringements against our Firebird brand.

The branding statement does a lot of footwork to make and reiterate the point that the product must be referred to as "Mozilla Firebird browser" and, preferably, as "Mozilla browser". It's not a "back-down" on the choice to take Firebird's brand-name. Still, now, provided all of the satellite websites comply AND "Firebird" disappears from all of their pages, documents and tags next month, as promised, then we can take comfort that the letter of the law, at least, is to be observed.

We wish that it had never become a question of confronting Mozilla. As a proposal, the name-change from "Phoenix" to "Firebird™" should never have gotten past the ethical questions raised in their forums and committees last December about taking another Open Source project's name.

We appreciate the attempts by Debian project admin Jonathan Walther, to try to mediate between us and the Mozilla folk. We look forward to some kind of contact from Mozilla's

leaders soon. We still have quite a lot to discuss with them. We trust that Mozilla will continue to do what is required to proactively cleanse the Web and their browser project of the brandname abuses, including the ™ claim on our Firebird mark.

Firebird users who would like to help Mozilla in this task can do so by posting any infringements you find to OUR Firebird-General list.

The Firebird Team

First Firebird Conference: In some sessions, it was "standing-room only"

By Helen Borrie for interbase-world.com

I'm sure everyone agrees that there was as much fun and spirit in the non-scheduled parts of the conference experience as there was in the programmed talks. Mark O'Donahue and I arrived on Saturday and did a bit of walking around in light drizzle. We found an Italian cafe (with the menu in German, natch) and had an interesting experience choosing lunch!

A little later, back at the hotel, we came upon a fellow who was trying to get some sense out of the charming but technophobic reception staff regarding connection to the hotel network for the Internet services none of us can live without.

As he wasn't German, he was speaking in English, clearly not his native language. I thought he might be French…well, we didn't get the network thing sorted at that point, and the fellow went away disappointed.

A while later (after I'd finally managed to connect to the 'Net using a borrowed ISDN-to-analog transponder) I got a call from Mark. "Hey", he said, "you know that guy in reception having trouble with his modem? It's Nickolay (Samofatov)!" Hmm. OK. (We got a bit less inhibited as time went on: I began approaching strangers and asking them if they were there for Firebird…a lot of them weren't…

Still in Saturday, the next highlight was dinner in an Indian restaurant (with the menu in Hindi this time, much easier!). It was a new experience for Nickolay, so we obligingly helped him by ordering Tandoori chicken and a sauce with a LOT of chilli. Well…I think Nickolay found that soft, black beer very soothing for his ravaged tastebuds!

Next morning, a few more souls drifted in. I met Pavel (who brought me a T-shirt from the Czech Republic, just a little too transparent to wear in public and stay decent) and Artur, who presented me with a beautiful CD of Portuguese singing. Around lunchtime we all went down to the hotel lounge bar and ordered beers and sandwiches. We made the fatal mistake of putting the sandwiches too close to Nickolay, so we had to order more.

The database talk never stopped. We chatted and held forth on every conceivable thing around what might be done, what was good, what was bad, made database jokes, got tongue-tied pronouncing names, et al. As people arrived, we added more tables and chairs, ordered

more beers, dragged more insights into the fair and fanciful database talk. It was just a great party. Eventually, groups split off and headed out in search of food and drink - not a simple task on a wet Sunday night in Fulda! But, judging by the sights next morning at 8 a.m. (registration time), they found some!

Monday 9 a.m. saw the official start of the conference. With the finale of Stravinsky's "Firebird" ballet suite as a fanfare, we had Sparky the mascot brought in on a tray. Then, starting with Holger (Klemt), who gave a preview and explained the technology, Frank (Ingermann) and I put each speaker on stage wearing a wolf mask. We got them to introduce themselves and their topics. Frank excelled himself by providing a running translation, for more than an hour, from English into German. Later, we had all the speakers sign one of the FirebirdSQL Foundation T-shirts using a wash-proof laundry marker. More about this T-shirt later….

Then the big room was partitioned off into three and the first sessions began.

From that moment forward, it was full-on. In some sessions, it was "standing-room only". I'm glad that Holger tape-recorded the sessions because I missed several that I really wanted to hear. We all provided transcripts of our slides and hand-outs, which Holger's amazingly wonderful staffers (Debbie and Andrea) burned onto CDs for us on Tuesday afternoon. I believe it is Holger's intention to make CDs available by subscription, including MP3 files of the tape-recorded sessions.

Ann's "Firebird internals and German beer" session on Monday night was a crack-up. While Ann valiantly tried to do her talk in an orderly fashion, using flip-charts, people were traipsing past in front of her to get their glasses refilled with a substance that resembled creamy-coloured marshmallow. Yeah - the hotel's bar staff didn't get the technology of the keg quite right, so the level of aeration was rather an insult to good German ale! It didn't stop anyone drinking it, however!

After that, the database talk spilled out all over town, with some very late technical sessions in the bars and cafes of Fulda. Tuesday morning saw three invalids delivering 8 a.m. sessions to VERY small audiences, enthusiastic queues for coffee at morning break and a notable lack of shaving. Life doesn't get better than this, does it? I have to admire Martijn for making it to his 8 a.m. talk, 'cause he looked REALLY sick!

Lunch on Tuesday was Italian - an excellent choice - because 24 people were wearing white T-shirts with Firebird logos that were perfectly colour-coordinated with the sauce.

We wrapped up the full conference on Tuesday evening with an auction to raise funds for the Foundation. This was a German "bridal auction", where bidding starts low, with the first bidder throwing his money into the bucket. After that, the bidding goes up by 2 or 5 Euros a time, each bidder throwing the increment into the bucket. Eventually the auctioneer - Mr Big-Hair Lucas (Franzen) goes outside and counts down from 10. When he stops, the last bidder wins the

auction.

In this manner, we got 183 Euros for the autographed T-shirt and 86 Euros for the Sparky mascot. Lucas twisted my arm and made me put my coffee plunger into the auction. He hammered me until I finally won it back - but not before I had thrown 10 Euros into the bucket! Still, this used $2.99 coffee plunger got us 33 Euros and gives me a wonderful excuse to plan a revenge for next year! Along with donations and the profits from selling Foundation T-shirts, we collected 500 Euros. Not bad at all!

Holger ran a lottery, with donated software products as prizes. Only those who had filled out evaluation sheets for the sessions were eligible. It was a very funny lottery, since it became clear that there were almost more prizes than there were people who had done the good deed with the sheets.

A big highlight for us of the Foundation committee was our first-ever "warm-body" committee meeting. Except for two (Geoff and Phil) we were all there.

In the photo we are (left to right) Svein Erling Tysvaer, Mark O'Donohue (president), Martijn Tonies, Sean Leyne, Lucas Franzen, Frank Ingermann, Helen Borrie, Thomas Steinmaurer and Jason Wharton.

There was one empty chair in every session - the one that Dmitry Yemanov should have been sitting in. It was a tragedy that our star was beaten by bureaucracy at the eleventh hour. We missed you, Dmitry, and there is NO WAY that you're going to miss next year's conference.

2004: Year of the Vulcan

The year started strong, as Firebird 1.5 was finally released on 21 February. The version was an instant success, especially because it brought the Classic architecture and the embedded server to Windows for the first time. With Classic available on Windows, the lack of SMP support in the SuperServer architecture no longer felt as critical to the community. And with a long-term solution promised through Project Vulcan, users were willing to wait even a few years. In the meantime, the embedded Windows server grew rapidly in popularity and was adopted in many consumer applications.

A week after the 1.5 release, the Firebird Foundation and HK-Software announced the second Firebird Conference, again scheduled for October in Fulda, Germany. With Jim Starkey among the speakers, it drew an even larger audience than the previous year.

While Jim worked on Vulcan, Firebird developers focused on Firebird 2.0. The Firebird-Architect mailing list was buzzing with traffic between Jim and the team, and for a while both projects seemed to be moving forward smoothly. But problems slowly emerged. Jim introduced a number of architectural changes in Vulcan, and not all of them were well received by the Firebird developers. The complex XML-style configuration system was one example. At the same time, Firebird had many bug-fixes that were never brought into Vulcan, and also absorbed enhancements from Yaffil. By December, when Jim completed the initial development phase, it had become clear that the two projects were drifting in different directions.

The Firebird community continued to grow stronger throughout the year. A great example of this momentum was the debut of the Fyracle project at the Firebird Conference in Fulda. Initiated and led by Paul Ruizendaal, Fyracle aimed to bring Oracle compatibility to Firebird—hence its nickname, "Oracle-mode Firebird." Its primary goal was to support Oracle-specific SQL syntax and PL/SQL, making it possible to run applications originally written for Oracle with little or no code changes. This dramatically reduced migration effort and lowered dependence on Oracle's proprietary ecosystem.

The pilot target was the open-source ERP/CRM system Compiere (the precursor of the ADempiere/Idempiere family), which relied heavily on Oracle PL/SQL. Fyracle was based on Firebird 1.5.1 and quickly gained traction; a beta was released in September, followed by version 0.8.0 in October.

Fyracle enjoyed considerable attention and reached peak adoption during 2005–2006, but development eventually stopped in 2007. But its legacy lives on, as many of its features and improvements found their way into Firebird 2.1.

Other notable events

The project was still using SourceForge's issue tracker, but it was becoming increasingly clear that a more capable solution was needed. After an evaluation of available options by Dimitrios Chr. Ioannidis, the team moved toward adopting JIRA.

Meanwhile, Adriano dos Santos Fernandes began work on a new internationalization architecture (INTL) for Firebird in a dedicated v2 branch in

CVS. This effort laid the foundation for the vastly improved Unicode and collation support introduced in later Firebird releases.

The Firebird Book written by Helen Borrie for IBPhoenix was published by APress, and quickly became the de facto “bible” for Firebird developers and administrators worldwide.

Global Context

While the Firebird community celebrated the 1.5 release, drew energy from the Vulcan and Fyracle developments, and enjoyed the success of the second international conference, a new competitor was rising rapidly on the global stage. Over the previous two years, PostgreSQL had gained significant press coverage, often portrayed as the most advanced and fastest-growing open-source RDBMS. Its adoption accelerated sharply, particularly in North America and Western Europe.

Firebird remained extremely strong in its core niches—the Delphi ecosystem, Eastern Europe, Latin America, and embedded applications—and it was technically competitive, even superior in many respects. Yet it lacked the broad public momentum and growing corporate backing that PostgreSQL was beginning to accumulate. The contrast between strong technical merit and limited global visibility would continue to shape Firebird’s strategic challenges in the years ahead.

2005: Trouble with Vulcan

The principal work on Vulcan ended, and code was handed over to SAS Institute for acceptance testing. The source tree was also pushed into project’s CVS as separate module. We cooperated with SAS on Vulcan testing and bug fixing. But the key question at that time was how to merge the Firebird 2 and Vulcan after a year of separate evolution. Architecturally, Vulcan was a huge departure from Firebird head, and the tree structure was almost completelly different.

The analysis done by Dmitry Yemanov concluded that merging Vulcan into Firebird 2 was feasible, but would require a long, difficult, and highly manual process. Both codebases had diverged too far for simple diff-based merging or commit replay, and neither direction—Vulcan→HEAD nor HEAD→Vulcan—offered a clear

advantage. Engine refactoring, new modules, incompatible class libraries, differing memory-management approaches, and inconsistent coding practices all ensured that no merge path would be quick or clean.

To address this, the proposed solution at the time was a third approach: creating a dedicated merge branch with an improved directory structure and integrating both trees step by step. This plan involved standardizing coding rules, unifying the class library, porting exceptions and configuration systems, adopting Vulcan's Y-valve and engine architecture, improving encapsulation and DSQL integration, and only then synchronizing the result with Firebird HEAD. Although more elaborate than a direct merge, this approach promised to combine Vulcan's architectural advances with Firebird's established codebase in a controlled, sustainable way. However, as Firebird 2 branch already accumulated number of significant fixes and enhancements and was ready for Alpha release, it meant that Firebird 2 will not address the SMP issue. The team decided to do not hold them for a merge that could take a year or two. The history shows that it was good decision, as it required much more time at the end.

The admin team also revisited the future versioning scheme, particularly in light of the complex Firebird/Vulcan integration. It was agreed that the first Firebird release with full SMP support would carry the 3.0 version number. At the same time, several significant features—such as the new INTL architecture and work on global temporary tables and CTEs (introduced in Fyracle)—were progressing independently. Rather than delay these advancements until the SMP-ready Firebird 3, the team decided to release them incrementally in the 2.x line. This led to the creation of Firebird 2.1 as the next step beyond 2.0.

The Firebird 2 Alpha 1 was released in March. This release contained a large number of new features, including derived tables, support for Execute Block, increased table sizes, new improved index code, expression indices, numerous optimiser improvements, enhanced security features, support for on-line incremental backups along with numerous other improvements and bug fixes.

While the new QA test suite was steadily improving and expanding, the sheer volume of new features in Firebird 2.0 meant that internal QA was constantly struggling to keep up. To strengthen testing as early as possible, the project issued a "Call for Testers", encouraging broad community participation from the very start

of the release cycle. Typically, most users began testing only once a version reached the Release Candidate stage—a pattern that had contributed to the long chain of RC builds during the 1.5 cycle.

In May, the Firebird Foundation announced a world-wide Firebird Conference to be held in November in Prague, Czech Republic. It became the largest Firebird event to date, at times featuring four parallel session tracks and keynote presentations in a grand lecture hall. A beta version of Firebird 2 was released shortly before the conference, and the Vulcan and its future merge with Firebird was one of the central topics presented and discussed by both the Firebird team and Jim Starkey, who was also among speakers. It was during this event when it was announced that the eventual unified codebase will be released as Firebird 3.

Meanwhile, Vulcan was also moving toward completion. IBPhoenix prepared it for release as a standalone open-source product for those who would like test it.

2006: Year of Two Birds

In late January 2006, the project prepared the second beta of Firebird 2.0. The CVS tree was tagged, the release notes were updated, and binary builds were being packaged. Although the beta had been planned for earlier, several issues required attention first, alongside the completion of the final 1.5.3 release.

The team announced that, assuming no critical problems surfaced, the next official build would be Release Candidate 1. From that point onward, only regression fixes would be accepted. Once RC1 shipped, developer focus was scheduled to shift toward the Vulcan merge and the new features and architectural improvements planned for Firebird 3.

In February, Jim Starkey announced that he had sold his Netfrastructure web-software business to MySQL AB and would be joining the company full-time. It later emerged that his primary mission there would be the development of a new transactional storage engine—**Falcon**—intended to compete directly with InnoDB.

Around the same time, SourceForge finally completed its rollout of Subversion support, and the Firebird project began preparing a migration from CVS to SVN. The transition proved far more complex than expected and ultimately took years to complete, in part because we (Dimitrios Ioannidis and me) first needed to finish

the move to the newly adopted JIRA issue tracker (As a side note: we also adapted JIRA 3 to use Firebird as its backend database).

At the end of March, the project published Firebird 2.0 Release Candidate 1. Despite the extensive field-testing programme, four additional RC builds were needed before the final release was ready. Firebird 2.0 was officially launched in November during the grand opening session of the fourth annual Firebird Conference in Prague, Czech Republic.

Version 2.0 marked a major milestone in Firebird's evolution—not only for its many new features and improvements, but also because it represented a clear break from its InterBase heritage. Although Firebird preserved compatibility with existing clients, most distributable files were renamed, and the new API and tooling adopted a distinctly Firebird-centric naming scheme.

Jim, Ann, and Netfrastructure

Message sent by Jim Starkey to firebird-devel list on Feb 18

My company, Netfrastructure, Inc., has been acquired by MySQL, AB. As part of the agreement, I will be working full time for MySQL. I expect to lurk on the architecture list from time to time and may contribute the occasional wolf-o-gram, but I will not be taking an active part in Firebird development. Although Ann will work for MySQL, part time, translating from wolf to English, she will continue to be active in the Firebird project.

My decision to join MySQL has almost nothing to do with Firebird and everything to do with Netfrastructure. The Netfrastructure platform represents what I feel about contemporary computing hardware and future application requirements, and has been the center of my technical heart and soul for six year. Some aspects of Netfrastructure technology have already been contributed to the Vulcan project, but Firebird and Netfrastructure are architecturally incompatible. An attempt to integrate the technologies would be unlikely to meet the goals of either project.

MySQL and Firebird have never seen each other as competitors and I doubt this will change in the future. The projects have different open source philosophies, different technologies, different customer bases, and different sweet spots. The ideas behind the two projects are, happily, public and available to all. If MySQL and Firebird compete, it is only competition in offering the best possible support to their respective customers.

I am pleased to have had the opportunity to finish the Vulcan project. The combination of Vulcan SMP and architecture combined the rich feature set of Firebird 2 will make a solid release and a superb platform for future development.

I wish the Firebird project all the best in years to come. And if you need an opinion, please feel free to call.

2007: The year lightning struck

Around the turn of the year, the project was hit by an unheralded shock: a third-party employer approached three of our key core developers with offers of full-time, salaried positions, intending to build its own fork of Firebird. To retain these essential people, the Foundation and three of our existing sponsors stepped in with immediate support. The intervention worked, but it came at a cost. For the Foundation, it meant shortening its projected funding runway from roughly twenty months to less than a year and facing the urgent need to reassess the project's long-term financial stability. From various sources we already knew that several companies had been eyeing Firebird's codeworkers as potential recruits. The events of January made painfully clear just how vulnerable a free-wheeling open-source project can be to commercial head-hunting.

On the development front, the year began with discussions on the firebird-devel mailing list about the detailed merge plan. Because the merge would require major rearrangements of the source tree, the long-planned transition to Subversion returned to the agenda with renewed urgency. Subversion was critical for preserving change history during such a large restructuring. Yet despite most Firebird subprojects—including QA and the .NET driver—having already migrated, the move of the core Firebird engine repositories remained incomplete throughout the year.

At that time the project was working across multiple Firebird versions. We were maintaining both the 1.5 and 2.0 lines, the HEAD branch contained ongoing work for 2.1, and there was the Vulcan tree and a staging area prepared for the future version 3 merge. Additional repositories supported drivers, QA tools, documentation, and other subsystems. Managing all of this was a genuinely heroic effort. Git, which would later make multi-branch workflows far easier, was still very new and only beginning to gain wider adoption. GitHub was in private beta as of April and would not open to the public until the following year.

Firebird 1.5.4 was released in February, followed by 2.0.1 in March together with Alpha release of 2.1. Then, in August, I received the following email:

I'm sorry that I have to inform you that David Stephen Rushby, 27, of Blue Grass, died Sunday, July 8, 2007, in a drowning accident in the Black Sea at Anapa, Russia. For last couple of years he was the sole developer and maintainer of the KinterbasDB Python connectivity package for Firebird and InterBase.

He was employed by Tander, Inc. in Krasnodar, Russia for the last nine months. He worked as project manger in the information technology division. He liked reading and writing, the study of foreign languages, cutting and splitting wood (that's where his SourceForge user name "woodsplitter" come from) and hiking. Most of all he enjoyed his work as computer consultant for Blue Grass Valley Bank and Page Valley Bank.

It struck me as lightning. Although David was not a member of the Firebird project, we had a very close partnership from the moment I began working on the QA system based on QMTest. I regularly reported issues with kinterbasdb, assisted with triage and testing, and even contributed some enhancements to the driver. His loss hit us hard—personally, and professionally as well. A critical part of our ecosystem was suddenly without a maintainer.

In July, Firebird won in two categories of the **SourceForge 2007 Community Choice Awards**: **Best Project for the Enterprise** (that came with nice $1000 donation to Foundation) and **Best User Support**. Around the same time, the Firebird 2.1 Beta was released.

The growing number of maintained Firebird versions was placing an increasingly heavy burden on quality assurance and other parts of the project. We were also dealing with recurring infrastructure issues that caused periodic outages of both the tracker and the website. Although these incidents were usually resolved quickly, they disrupted the workflow and added to the sense of strain.

In October, the International Firebird Conference returned under HK-Software's banner, this time in Hamburg, Germany. Alongside the technical sessions, discussions about Firebird's future dominated the event. They took place everywhere: between sessions, in the lobby, and in a dedicated "all hands" session. Similar conversations had already been ongoing within the Foundation for months.

By then, it had become clear that a fully multi-threaded SuperServer with shared cache could not be delivered the following year. Based on the progress of the merge process, the developers agreed to produce an intermediate release instead, based on the transition branch. This version would support a multi-threaded, non-shared-cache mode—essentially the setup used in SAS production systems and known as SuperClassic at RedSoft—and introduce a new ODS. This interim release became known under the codename **Firebird 2.5**.

Toward the end of the year, as the “Future for Firebird” discussions on the FFMembers list tapered off, the Foundation Committee recognized the broad sentiment that some form of structural change was needed. They convened a **Technical Task Group (TTG)** made up, by invitation, of Firebird project administrators and major sponsors. The committee appointed Roman Rokytskyy as its representative, while the general membership elected Frank Ingermann and Martijn Tonies. The TTG then appointed me as its chairman and Roman as secretary.

Other notable events

In January, Sun Microsystems Announced agreement to acquire MySQL.

New Firebird header on main page introduced in 2007

2008: Finding a way to the Future

The year started with first TTG meeting. Initial agendas were prepared with the aim of getting the group members acquainted with the project’s worldview, especially its Roadmap of feature sets for forthcoming releases. The sponsors and members were expected to express views of the proposals and to have some degree of influence on the shape of the final Roadmap. For a few weeks, these matters were discussed, with the project’s coordinator, Dmitry Yemanov, presenting a tentative roadmap for v.2.1 and 2.5 as fuel for discussion. The Roadmap that emerged was both realistic and encouraging.

In April, we released Firebird 2.1 containing many sought-after new features including database monitoring, global temporary tables, Common Table Expressions, database triggers, SQL extensions, and dozens of new internal functions. During the year, we released also maintenance versions to both 1.5 and 2.1 lines, and two Beta versions of Firebird 2.5.

In September, Fabio Codebue and his colleagues staged the sixth International

Firebird Conference in Bergamo, Italy. The Foundation was not involved in the organisation but assisted with the selection of papers and the organisation and publication of the programme.

The global financial crisis of 2008 also took its toll on the Firebird Foundation. By the end of the year, the situation had become alarming. The project was supporting and developing four active release branches, yet the Foundation had nowhere near enough funds to expand the pool of core developers and testers or even maintain it. At the prevailing pace, it seemed likely that some of our existing contributors would be forced to seek other work by Easter 2009. When the Treasurer and other committee members raised these concerns on the FFMembers list, several ideas circulated, though none led to immediate structural change. What did happen was a groundswell of support from within the community. Existing sponsorships were renewed and, in some cases, increased. Several Foundation members recruited new sponsors or became sponsors themselves. A membership campaign led by Carlos Cantu in Brazil brought in a substantial number of new Foundation members in October and November, and overall renewals were strong. To ease financial pressure, both Philippe Makowski and I voluntarily relinquished our QA grants.

But alongside the financial difficulties, we also faced serious technical challenges. With no maintainer, the kinterbasdb driver quickly began to fall behind Firebird's development, putting the entire QA process at risk. Certain new features could not be tested at all, and the gap widened steadily. When no one in the community stepped forward to continue David's work, we had to step in ourselves. kinterbasdb, being a C++ extension for Python, came with a steep learning curve, but we eventually managed to update the driver (version 3.2.2) sufficiently to support the Firebird 2.1 QA process. Even so, it was clear from the outset that this approach was not sustainable.

Other notable events

Embarcadero, after acquiring Borland's CodeGear division, continued to market InterBase as a commercial product. Yet signs soon appeared—in blogs and even in a few press releases—that the company had begun incorporating Firebird support into its development tools. The anti-Firebird sentiment that had lingered from the old Borland years gradually gave way to a more positive stance, reflecting

Embarcadero's recognition that many Delphi and other former Borland-tool users were active members of the Firebird community. Judging by the comments in our forums at the time, this shift was welcomed by a large part of the Firebird user base.

In mid-June, Jim Starkey left MySQL AB to launch a new venture, NimbusDB (later NuoDB), bringing his work on the Falcon storage engine to an end.

2009: Back in the saddle

The whole year was devoted to intensive work on Firebrid 2.5, and maintenance of three older lineages. To relieve some burden on development team, it was decided that 1.5.6 release would be the last one in 1.5 line.

In October, Dmitry Yemanov presented a comprehensive development strategy that shaped much of the project's direction in the years that followed. His proposal outlined the major goals for version 3: a scalable SuperServer architecture, deep internal refactoring, and a reworked security model. One of the key recommendations was to split the notion of a "shared cache" into two components —a shared page cache, considered essential for Firebird 3, and a metadata cache that could remain per-attachment due to complexity and stability concerns. This approach avoided the synchronization problems encountered in Vulcan and set the foundation for the unified, multi-architecture engine design that Firebird eventually adopted.

Dmitry also proposed significant architectural clean-up: eliminating the old dbb_sync mechanism, moving nonshared structures from the Database level to the Attachment level, replacing compile-time SUPERSERVER/CLASSIC switches with a runtime configuration option, and refactoring the Y-valve while keeping the system simple to configure. Although Vulcan used multiple client libraries, Firebird decided to retain a single client library for ease of deployment. Likewise, instead of adopting Vulcan's configuration system wholesale, the project opted for a Vulcan-inspired design that preserved its flexibility without replicating its complexity. The plan also called for moving authentication to the remote subsystem, adding a configurable embedded-user authentication option, and reviewing RedSoft's security patches for possible inclusion.

Beyond architecture and security, the proposal sketched out an incremental path for SQL and optimizer improvements and set boundaries for public API changes. Dmitry recommended a feature-based Alpha milestone and a schedule-based Beta, with any unfinished work deferred to later versions to maintain release discipline. This strategy, widely endorsed in the subsequent discussion, provided the roadmap for the Firebird 3 development cycle and guided the project through the complex Firebird/Vulcan merge that followed.

In July, Firebird earned another SourceForge Community Choice Award, this time "only" in the **Best Project for Enterprise** category.

Another major achievement of the year was Firebird's official acceptance into all major Linux distributions. Although Firebird had long been available through various "contrib" or community repositories, it had been officially provided only by Debian and Mandrake. In 2009, Firebird finally entered the official repositories of OpenSuSE, Red Hat/CentOS/Fedora, and others. Acceptance into the Red Hat ecosystem was especially significant, as Red Hat had traditionally been seen as a stronghold of PostgreSQL.

In December, the first Release Candidate of Firebird 2.5 was published. Not long afterward, a disk controller failure severely crippled the Firebird infrastructure hosted at BroadView Software. The Firebird Wiki VM was lost completely, and both the tracker and the website suffered heavy damage. While backups allowed basic services to be restored quickly, it took weeks to fully rebuild the systems, especially the tracker, and some content was lost permanently.

The `kinterbasdb` saga continued with the 3.3.0 release, which enabled basic QA testing for Firebird 2.5. However, we lacked the resources to extend the driver to cover all the new features introduced in that version. As a result, I decided to develop a new Firebird driver in pure Python, built from scratch as a seamless replacement. Early prototypes that used the ctypes module to interface with the Firebird client library showed real promise, and work on what became the **FDB** driver began.

Other notable events

In April, Oracle announced it will buy Sun Microsystems, and MySQL with it. The deal was finished in January 2010, and because Oracle aquired InnoDB back in

2005, it basically meant the death sentence for the Falcon engine.

In May, the last shreds of the once-mighty Borland have gone under the hammer. It has been bought by a U.K. company MicroFocus International PLC that produced software for testing software. The Codegear, the arm of Borland that maintained InterBase and produced Delphi, went to Embarcadero Technologies previous year.

2010: Mind the Bird!

The year was shaped by the synergy of two major milestones: the upcoming 10th anniversary of the project in July and the long-awaited final release of Firebird 2.5. Everything revolved around these goals, and even the annual Firebird conference was intentionally skipped to avoid any distraction from delivering 2.5 on time.

Instead, the community launched the **Mind the Bird!** campaign, led by Dmitry Kouzmenko, to raise Firebird's public profile as the new release approached. As part of the effort, Carlos Cantu announced a Firebird 2.5 article-writing contest with cash prizes. Although there wasn't International Conference, several local community events were organized as part of the campaign. **MindTheBird!** became one of the most successful promotional initiatives in Firebird's history, generating wide visibility and renewed enthusiasm.

2011: The beginning of the long march

Until this point, the project had managed to release a new major version every even-numbered year, while maintaining older branches along the way. Naturally, expectations were the same for Firebird 3: an Alpha and Beta in 2011, followed by the final release the next year. But as the year unfolded, it became clear that Firebird 3 would break this pattern. The sheer volume of changes pushed the schedule far beyond anything seen before.

What began as a straightforward "merge" of Vulcan architecture into the Firebird codebase evolved into a full-scale overhaul of the engine. Virtually every subsystem was touched, redesigned, or replaced as developers worked to build the strongest possible foundation for all future versions. New memory management, new synchronization primitives, an object-oriented client API, an internal plugin architecture, a completely reworked security model, and other deep architectural

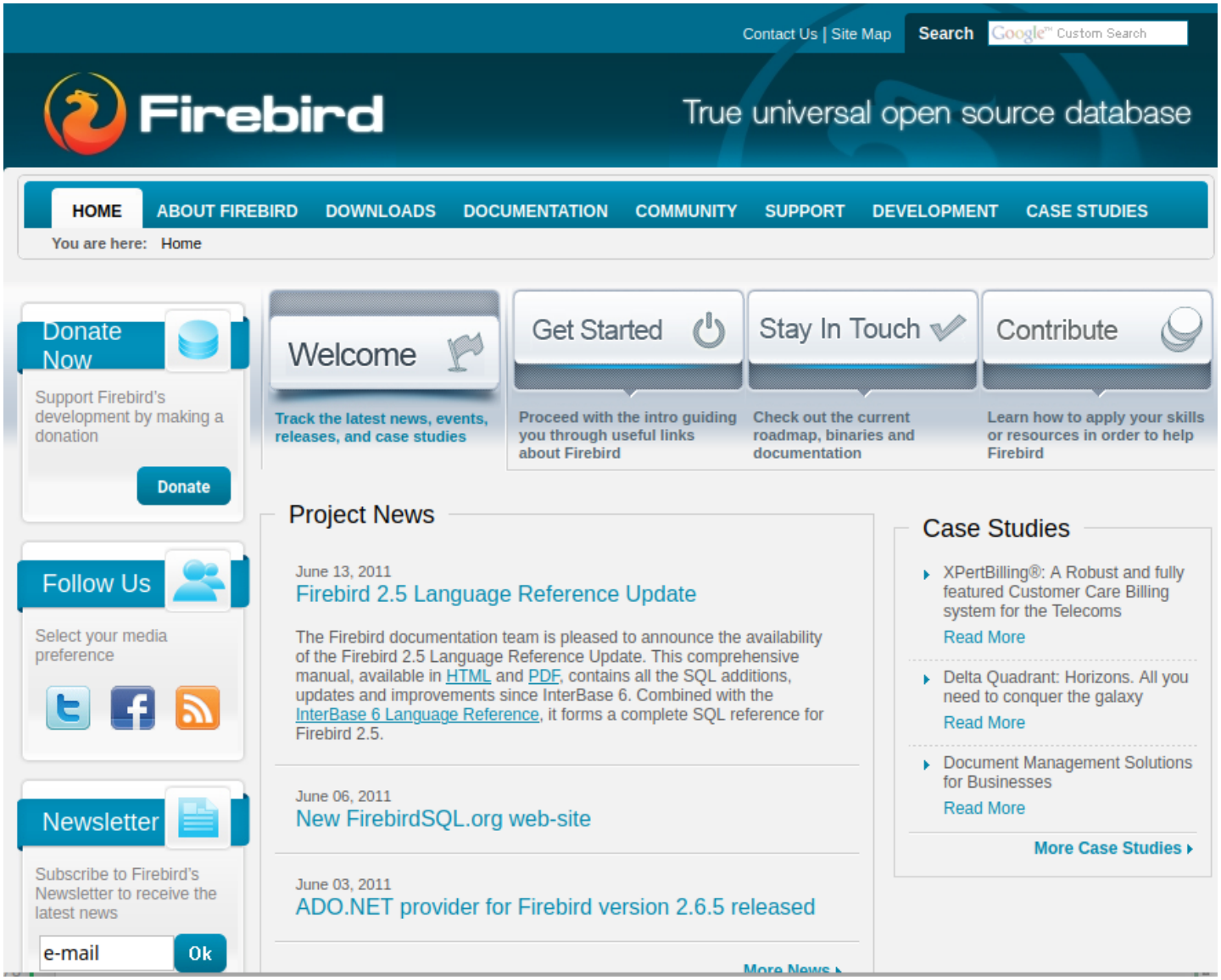
changes all began landing in the tree. At the same time, the core team was still maintaining three older release lines.

Unsurprisingly, the Firebird 3 Alpha slipped further and further into the future, taking the final release with it. Looking back, 2011 marked the start of a long march that would continue for six full years.

MindTheBird! website

Alongside work on the Firebird engine, the project was also busy with several auxiliary efforts. The Firebird website was migrated from its old, unreliable hosting setup to three VPS instances spread across two separate data centers. It also underwent a complete redesign, with all content moved from the custom PHP site to a new CMS provided by DQTeam.

New Firebird website introduced in 2011



The long-overdue migration of the Firebird source code from CVS to Subversion was finally completed in the summer. It was a gargantuan effort that pushed both the hardware and the capabilities of the available tooling to their limits. The central repository, ***firebird2***, was many gigabytes in size, and converting it to the new format alone took more than six hours. The process was far from smooth at first; the conversion tools repeatedly choked on the sheer volume of data and certain oddities in tags. Once the conversion succeeded, the new repository had to be uploaded to SourceForge, installed into their infrastructure, and equipped with all the necessary commit hooks. To minimize disruption for developers, the entire procedure was rehearsed multiple times to ensure that the final cutover would go smoothly and quickly as possible.

In November, the ninth Firebird Conference was hosted by Sita Software, a Luxembourg-based company specializing in ERP, accounting, and business management solutions. As in previous years, the event combined core team presentations, third-party sessions, and hands-on discussions. It gave a noticeable boost to the adoption of Firebird 2.5 and helped shape the direction of ongoing driver and tooling development.

In December, the initial version of the FDB driver was released. Although not yet feature complete, it was stable enough to start his integration into the Firebird QA process.

2012: Business as usual

This year was one of the quietest and most uneventful in Firebird's history, with the sole exception of the annual Firebird Conference. As in the previous year, it was hosted by Sita Software in Luxembourg, this time in October, and once again it stood out thanks to the hospitality of Vincenzo Sita and Jas Madhur. The atmosphere felt more like a family reunion with technical sessions than a typical IT conference.

Meanwhile, the project continued its steady work on Firebird 3 and released maintenance updates for all supported branches, following the policy of delivering at least one per year for each. The drivers developed under the Firebird umbrella also continued to evolve throughout the year.

To reduce the burden on the core team, the project decided to limit support to just two active lineages: the latest release and the one preceding it. As a result, a final maintenance update for the 2.0 series was published. This maintenance policy remained in place for many years, up to the release of Firebird 5.0, which should have marked the end of the 3.0 line. However, because a significant number of deployments had not yet upgraded, the decision was made to extend 3.0's life for a few more years.

2013: Slow progress

In February, we learned with great sadness of the passing of Roger Vellacott, who had died of cancer the previous November. Roger had been an important

contributor to Firebird and its community from the earliest days: a voting member of the Firebird Foundation, a steady presence in the support forums, and the head of Passfield Data Systems in Bath, U.K., a long-time Firebird sponsor.

March brought two pieces of good news. An expanded Second Edition of Helen Borrie's excellent The Firebird Book was published, this time split into three volumes available in both print and PDF formats. And the FDB driver reached full maturity with the release of version 1.0. It was a prerequisite for complete overhaul of our QA toolchain. The QMTest was replaced with homegrown system called fbtest.

In August, the first Alpha of Firebird 3.0 was finally released. As an Alpha, it was not feature complete, but it offered the first real glimpse into the shape of the next-generation engine.

A major milestone followed in July, when Firebird Embedded and its corresponding driver were merged into the LibreOffice master Git branch. Becoming the storage engine for LibreBase (although optional as Java-based HSQLDB was still default) was a significant validation of Firebird Embedded's reliability and global reach. While such deployments rarely increase Firebird's public visibility—end users typically have no idea that Firebird is managing their data—they greatly strengthen Firebird's profile and credibility among application developers.

There was no conference that year, but IBSurgeon, together with several core developers, organized a technical Firebird roadshow across Europe. These sessions brought Firebird expertise directly to local communities and helped maintain momentum despite the absence of a formal international conference.

2014: Are we there yet?

A second Alpha of Firebird 3 was released in February, bringing another wave of new features. The International Firebird Conference returned to Prague that October. As always, it was a lively event, but the atmosphere had changed. The excitement that defined earlier conferences was now mixed with a growing unease. The final release of Firebird 3 was already two years behind the original expectation, and patience was wearing thin.

Despite the delay, Firebird 2.5 had taken deep root worldwide since its release. The user community had, in a sense, made peace with the situation and focused its efforts and investments around 2.5, which had proven to be a rock-solid foundation. The ecosystem around it expanded steadily over the previous four years: new applications and deployments, new tools and drivers, and even consumer products built on top of Firebird. The engine also began its first steps toward Android and iOS through early experimental ports.

So when the Beta version of Firebird 3 was finally published on 1 December, it arrived quietly. No fanfare, no celebration—just cautious progress.

Of course, throughout the year the project remained busy across a wide range of tasks, producing many tangible results for Firebird users. The Documentation team continued releasing new manuals and guides. The QA team expanded the test suite, now fully leveraging the FDB driver to exercise every aspect of the engine. All drivers maintained by project members saw steady improvements, and both the Firebird 2.1 and 2.5 lines received new maintenance releases.

2015: Almost there, but not

Adopting new tools within the Firebird project had always been a slow and careful process. We had to weigh functionality, platform constraints, learning curves, migration difficulty, and personal preferences. Any change that risked disrupting development tended to meet resistance and was often postponed to “quieter” times. Still, discussions about adopting new compilers, version control systems, issue trackers, or CMS platforms were a constant part of project life, resurfacing regularly over the years. Some migrations were long-planned, like the moves to JIRA and Subversion; others happened more spontaneously. The adoption of Git and the gradual migration from SourceForge to GitHub fell into the latter category.

It began when Adriano created a personal clone of the Firebird source tree on GitHub. Then, in February, Jiří Činčura established the FirebirdSQL organization there, sparking discussion among the project admins about a possible migration. As with the earlier move to Subversion, the transition unfolded gradually, with individual projects moving at their own pace. The Jaybird was the first to migrate in April, while the core Firebird repositories remained on SourceForge—developers preferred to spend their time on code rather than tooling transitions.

In late July, SourceForge suffered an incident that affected our Subversion repositories, forcing a restore from backups and resulting in the loss of about 1.5 days of work on the Firebird engine. Combined with SourceForge's recent turn toward aggressive advertising on download pages, the incident became the final straw. An exit plan was drafted, with the goal of transitioning to GitHub in the autumn, once Firebird 3 entered its final pre-release phase.

The move, however, was far from simple, and took longer than envisioned. SourceForge also hosted our mailing lists, a feature GitHub did not provided, so alternatives had to be found. Then there was the JIRA tracker: hosted at BroadView, still plagued by periodic outages, and stuck on the aging 3.13 version. We could not easily upgrade it because it had been adapted to use Firebird as its backend and customized with additional fields and workflows. Switching to GitHub's integrated issue tracker made sense in the context of the migration, but it added yet another layer of complexity to the already delicate transition.

There was also discussion among the admins about holding a Firebird Conference, but it was agreed that without the final release of Firebird 3, such an event would distract developers and offer limited value to attendees. Instead, another Firebird Tour across Europe was organized by IBSurgeon and IBPhoenix, with support from the core team.

Once again, the Firebird Foundation faced serious financial pressure to keep the project running. An emergency appeal was issued to Firebird users, and the community responded with remarkable generosity, helping stabilize the situation.

On 9 November, Firebird 3 Release Candidate 1 was published.

2016: Firebird 3, at last! But where are the cheers?

It was a testament to the strength of the refined QA process and the dedication of the core team that such a major release required only two Release Candidates before Firebird 3 was officially published on 19 April.

Although the new version was welcomed with genuine gratitude, its adoption rate during the year remained surprisingly low. The main challenge was the number of significant changes introduced in Firebird 3: a new configuration model, a redesigned security system, new optimization mechanisms, and many internal

architectural differences. Unlike previous upgrades, moving to Firebird 3 was not simply a matter of reinstalling the server and performing a backup and restore with gbak. Applications themselves needed to be tested carefully, because the new engine behaved slightly differently in several operational areas.

This extra effort—varying from modest to substantial depending on the application—slowed down adoption across the user base. Over time, this phenomenon became known as "The Great Upgrade Barrier".

The situation was not helped by occasional reports of query performance regressions in specific cases. Although every issue was promptly investigated and resolved, it still took nearly two years before Firebird 3 gained full acceptance among users and saw its deployment numbers rise significantly. Even then, the growth came largely from new installations and applications rather than from upgrades; migration from version 2.5 remained relatively low.

The Firebird Conference returned to Prague in October. Unsurprisingly, the central themes were Firebird 3 and the forward-looking plans for Firebird 4 and the project as a whole. The atmosphere was positive, even enthusiastic at times, but it carried a trace of bitterness. Fatigue and uncertainty from the ongoing challenges facing both the project and the Foundation were clearly noticeable. The lingering effects of the 2008 financial crisis, along with natural business shifts, had taken a toll on many long-time sponsors. Some companies changed management, others closed, and some were acquired and forced to adopt different technology stacks. Whatever the reason, the outcome was the same: the number of Foundation sponsors continued to decline.

At the same time, attracting new sponsors became increasingly difficult. This was not because Firebird's user base was shrinking—quite the opposite. Global deployments had grown markedly since the release of version 2.5. But attitudes toward supporting open-source projects had shifted over the years, and fewer companies were willing to provide direct financial backing, even for software they used extensively.

But the work continued as before. Both the 2.5 and 3.0 lines received maintenance releases, while the HEAD branch opened fully for new Firebird 4 development. The tracker continued to suffer periodic outages, and the migration effort moved slowly but steadily forward. The mailing lists found a new home on Google Groups,

and individual projects — including the Firebird engine — were gradually migrated to GitHub.

After the Firebird 3 release schedule “disaster”, we were determined to return to the pattern of delivering a major version every two years. At the conference, Dmitry Yemanov presented a roadmap for Firebird 4, with the final release expected by the end of 2017. We began dividing planned features into mandatory items that had to be included in the version and optional items that could be deferred if deadlines became tight. Several mandatory features — built-in logical row-level replication, metadata identifiers longer than 31 characters, statement and connection timeouts, and predefined system roles — were already mostly implemented or close to completion. The remaining major items were batch API operations and high-precision numerics beyond 18 digits. And there were also some minor features and improvements. At the time, we believed this plan was achievable within the proposed schedule. But we were wrong…

Other notable events

In May, the project received the sad news that Aage Johansen, one of the earliest and most steadfast members of the Firebird community, had passed away in Oslo. Aage was a founding member of the Firebird Foundation and a loyal supporter of the project from its beginnings in 1999. He contributed quietly but generously over the years, assisting on support lists, reviewing documentation, and testing international character set issues. His colleague at Kreftregisteret Norway, Svein Erling Tysvaer, noted that Aage would be deeply missed, both personally and professionally, as he was the only person combining medical, programming, and historical expertise—skills that made parts of his work irreplaceable. Those of us in the Firebird project who had the privilege of working with him felt the same.

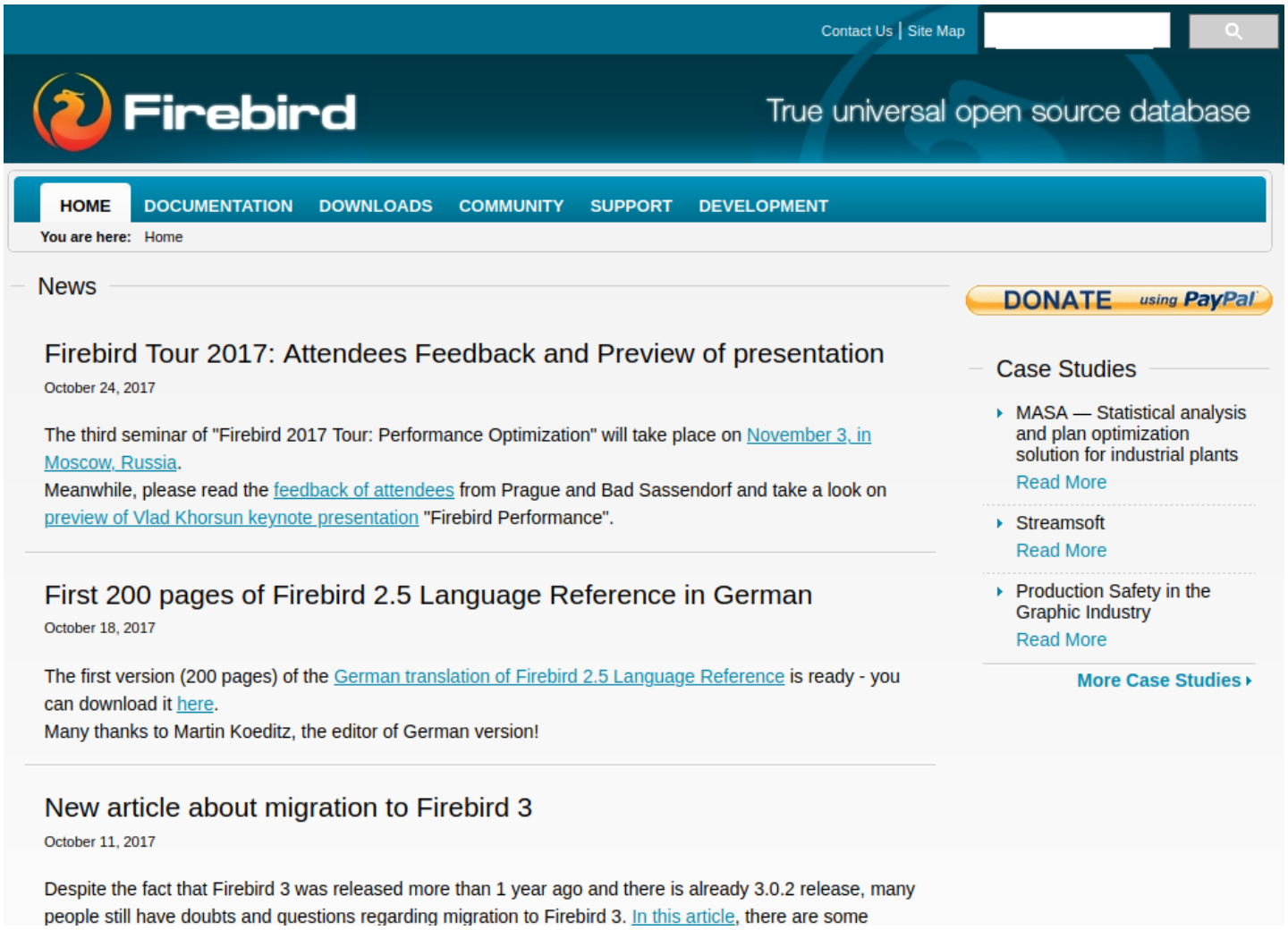
At the end of the year, the LibreOffice team announced that Firebird 3 would become the **default** storage engine for LibreOffice 5.4. In practice, however, the change required more time to mature and ultimately materialized in version 6.1, released in 2018.

Firebird was reset to "optional" status in early 2022, reverting HSQLDB as the default. Despite this, Firebird remained fully functional and was not deprecated—community clarifications emphasized it was "destined for no removal."

2017: The beginning of a new era

With Firebird 3, the project entered a new era. Its modular yet compact architecture preserved all the qualities users valued while establishing a solid, modern foundation for future development. Some advancements moved directly into Firebird 4, such as built-in replication and new data types like DECFLOAT and INT128. Others emerged as separate open-source or commercial additions, including external engines for writing stored procedures in .NET or Java, and various encryption plugins.

Firebird website adjustments in 2017



Inside the project, daily work continued as usual, but all processes and workflows were gradually adapted to the project's new home on GitHub. The broader ecosystem evolved in parallel. Open-source tools and commercial products surrounding Firebird steadily added support for Firebird 3, enabling users to take

advantage of its expanded capabilities.

Documentation also advanced significantly. Helen Borrie's *The Firebird Book* gained an additional volume dedicated to Firebird 3. In April, the project launched a fundraising campaign to translate the *Firebird Developer's Guide* from Russian to English. Written primarily by Denis Simonov in 2015–2016 and sponsored by IBSurgeon and the Moscow Exchange, the guide provided step-by-step examples for new Firebird developers, complete with full source code and parallel examples in Delphi, .NET desktop, .NET MVC, PHP (Laravel), and Java (Jaybird 2.x), with databases in both 2.5 and 3.0 formats. The campaign succeeded, reaching its 5,000 USD goal in August.

In November, the Firebird 2.5 Language Reference was completed. At more than 500 pages, it was a major achievement. Work had begun in 2015 with the translation of the original Russian edition, and because Firebird's language evolves incrementally, this manuscript became the foundation for all future Language Reference volumes.

As in the previous year, the traditional conference was replaced by a new Firebird Tour across Europe.

Throughout the year, it became increasingly clear that we could not meet the ambitious goal of releasing Firebird 4 by the end of 2017. The Alpha was not ready until August, already too late for the planned schedule. What we underestimated was the workload generated by maintaining the other release lines, especially Firebird 3. As adoption finally gained momentum, users uncovered issues in specific real-world environments that had not appeared during internal testing. Fixing these promptly took priority over everything else. Even so, we still believed the final release would be delayed by no more than a year.

2018: Second wind

For the project, 2018 was largely business as usual. In October, Firebird 3 finally reached full maturity with the 3.0.4 maintenance release, allowing the team to shift its main focus to version 4. The core team aimed to publish the first Firebird 4 Beta by the end of the year, and all preparations were in place for a Christmas release. In the end, a small hiccup in the internal process pushed the release to February of the following year. Still, compared with the earlier turbulence,

progress was smoother and more predictable.

After years of operating on the edge, the Firebird Foundation also experienced a welcome shift toward stability. Increased support from sponsors, members, and donors helped improve both the financial situation and the long-term outlook. For the first time in several years, the Foundation no longer felt like it was fighting simply to survive.

At the same time, with the possibility of another delayed major release looming, discussion began about how to strengthen the project's visibility and strategic position. In the autumn, IBPhoenix initiated a conversation about expanding Firebird's scope beyond the engine and core connectivity layers into a new domain: administration and management tools, along with integrated solutions for enterprise IT systems. The initiative that emerged from these discussions became known as **Firebird Butler**, envisioned as a new, independent division within the project. Preparations began for its formal launch in the following year.

Other notable events

In March, the English edition of the *Firebird 3 Developer's Guide* was published. And as in previous years, the traditional international conference was replaced by another Firebird Tour across Europe.

2019: Berlin! Berlin!

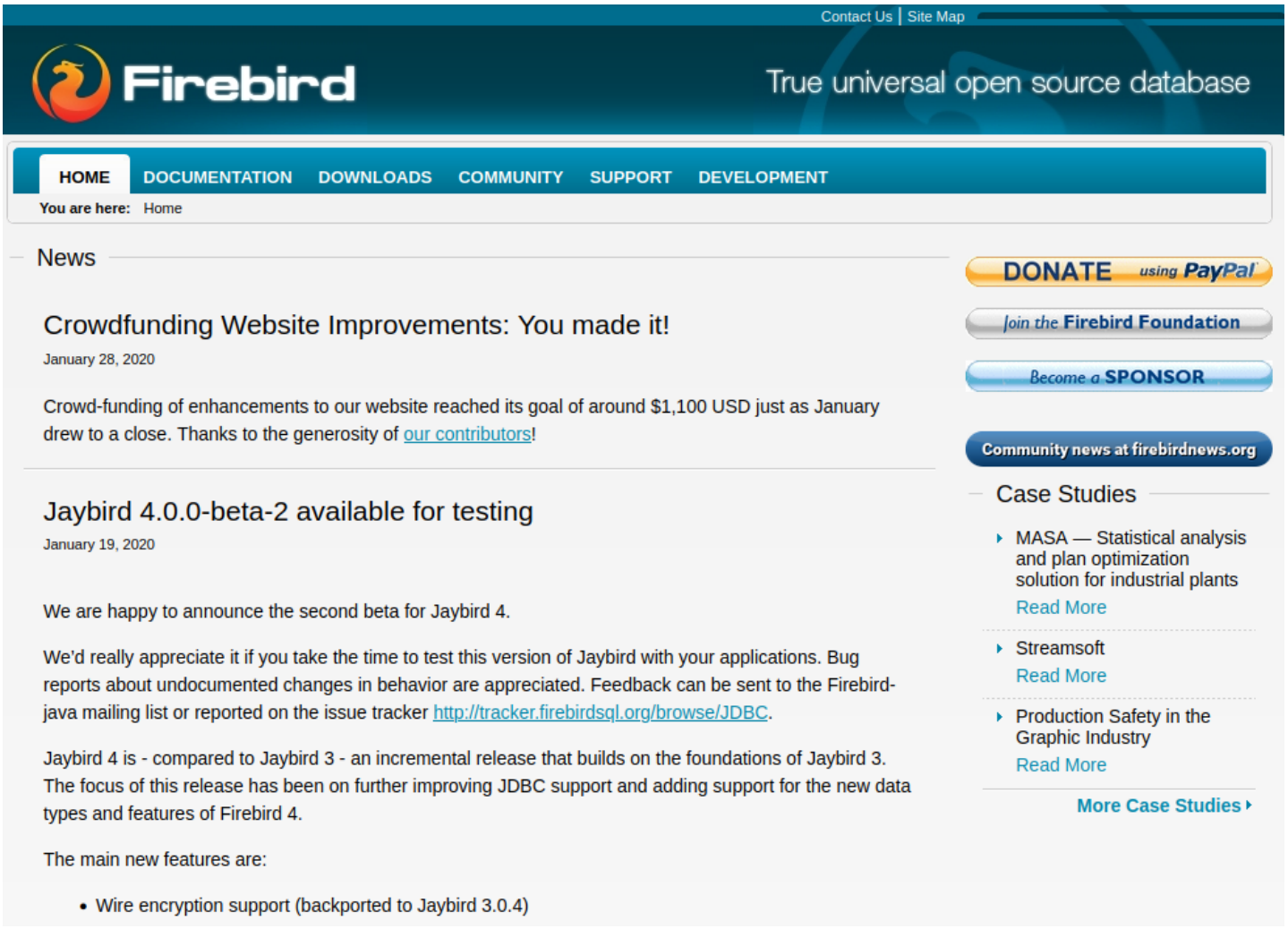
The year began with preparations for the Firebird 4 Beta release, the launch of Firebird Butler, and the announcement of the International Firebird Conference planned for October. All three announcements went live in February, marking the start of an energetic and optimistic year.

The conference was scheduled to begin on October 17, and all efforts were focused on making that deadline meaningful. Our goal was to present tangible results of the work completed so far along with a realistic roadmap for what lay ahead. As always, the program featured a mix of practical workshops and sessions covering a wide range of topics. The event proved to be a great success, and the enthusiastic atmosphere of earlier conferences returned in full. The most common farewell at the end was, "See you next year!" If only we had known…

2020: COVID here, COVID there

The COVID outbreak slowed us down, but the work continued. Thankfully, we did not experience any losses within the project or the Foundation.

In May, we released the second Beta of Firebird 4, along with the first version of a completely new generation of the Python driver, as announced at the Berlin conference. Development of **Saturnin** — the reference implementation of the Firebird Butler standards and development platform — had reached the point where some services interacting with the Firebird server were already functioning. However, the old FDB driver had become a serious obstacle. Written originally for Python 2, with later Python 3 compatibility layered on top, it no longer aligned with Saturnin's modern Python 3 foundation. Clearing this technical debt became essential, so a new Python driver was created, built directly on Firebird's OO API.



Once the new Python driver was basically completed in September, focus shifted to our QA test harness, *fbtest*. It too needed a transition to Python 3, and FDB had to be replaced with the new driver to allow tests to access Firebird 4 features such as new data types and time zones. But as we explored the upgrade path, it became clear that updating fbtest would be more complicated than rewriting it entirely. With roughly two thousand tests already in place, switching to a completely new system was unrealistic. Instead, we decided to build a modern replacement for fbtest that would make migrating the existing tests as painless as possible. Because this required several months of concentrated work, the effort was scheduled for the following year, and the Foundation was asked to set aside funds to support it.

Meanwhile, as Yahoo Groups shut down after many years of hosting Firebird-related community discussions, we migrated all remaining groups to Google Groups as well.

On New Year's Eve, the Firebird 4 source tree was finally tagged as Release Candidate 1.

2021: Would you like Firebird 4?

The path to the Firebird 4 final release was pleasantly straightforward. The Release Candidate was published on 1 February, and after a few months without any serious issues reported, it was rebuilt and officially released on 1 June.

In the meantime, the entire issue tracker was migrated from JIRA to GitHub in April, completing our transition to GitHub infrastructure.

The documentation team also had a productive year, publishing the *Firebird 3 Language Reference* in February and the *Firebird 4 Language Reference* in December.

And in May, work officially began on the next major version: **Firebird 5**.

2022: Nothing to see here, move along

Work on Firebird 5 continued steadily, along with maintenance releases for the 3.0 and 4.0 lines, ongoing driver development, and updates to the documentation. A major effort this year was the creation of a new test system built as an extension to the Pytest framework, followed by the conversion of all existing tests to the new

format. Although fbtest had used a declarative approach and Pytest relies on Python functions in traditional source files, we developed a conversion tool that automated much of the migration and greatly simplified the task. Most tests worked after conversion, but several hundred required manual adjustments or a full rewrite. This effort occupied much of the year, as we also expanded and refined the Pytest plugin to meet new requirements as they emerged.

2023: Objects in Motion

The principal work on Firebird 5 was largely completed, and the Beta preview was published in March. With the QA overhaul mostly finished, development on the Saturnin platform and its core services resumed, leading to the release of version 0.8 in April. After issuing routine maintenance releases for the 3.x and 4.x lines, the team was ready to finalize Firebird 5. The first Release Candidate appeared in September, but a few remaining issues required a second RC in December, pushing the final release into the following year.

The project website also received a long-awaited facelift, making it more attractive, easier to navigate, and fully mobile-friendly.

Despite the challenging global economic climate marked by rising inflation and instability in many countries, the Firebird Foundation's financial situation remained stable thanks to the continued support of sponsors, members, and donors. However, Helen Borrie's ongoing health problems forced us to confront a difficult question: whether the Foundation should remain based in Australia. With most of the active membership now located in Europe, the committee began evaluating options for relocation. After careful consideration, it was decided to move the Foundation to the Czech Republic during the coming year, and the necessary legal and administrative steps were set in motion.

2024: Born Again

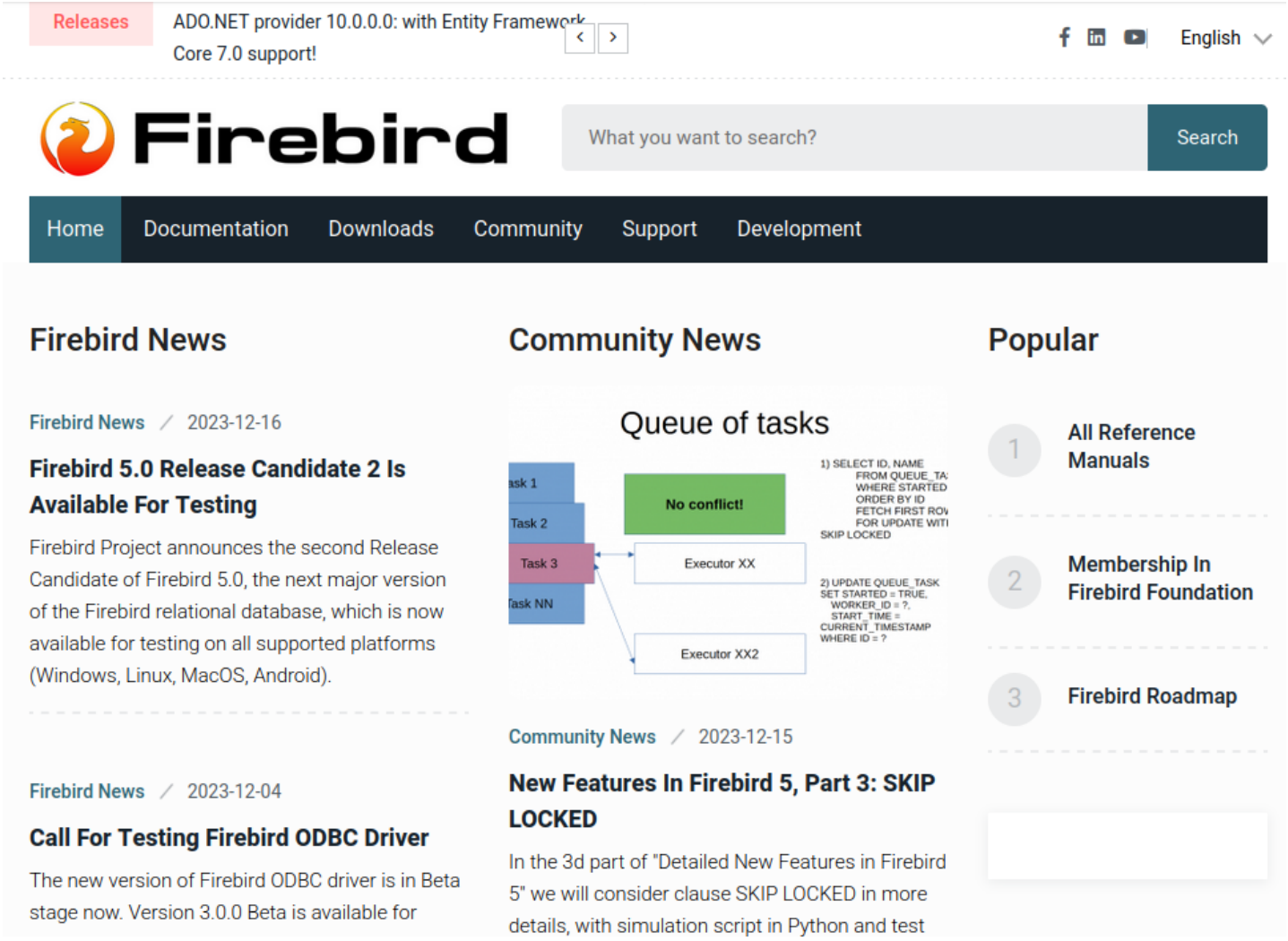
The year opened with the release of Firebird 5 in January. Soon after, we convened another Firebird Technical Task Group (TTG) meeting to finalize the development plan for version 6, and an updated Firebird Roadmap was published in March. While the project itself moved forward in its usual steady rhythm, the Foundation

entered one of its busiest years in decades as it prepared for its transition to a new home in the Czech Republic.

In October, the Firebird Foundation Inc. in Australia held its final General Meeting, during which members voted to dissolve the organization and pass the baton to its successor: Firebird Foundation z.s. A founding General Meeting for the new entity followed in November, establishing new processes, infrastructure, and plans for the years ahead. In a very real sense, the Foundation was reborn — in true Firebird fashion.

As a harbinger of the new era, the first exam under the Foundation's certification program took place in November, producing the very first group of certified Firebird professionals. And in December, the inaugural issue of this magazine was published under the Foundation's banner.

Rejuvenated Firebird website, December 2023



2025: Neverending Story

And here we are, in the final year covered by this 25-year history of Firebird. Sadly, it began with heartbreaking news: our dear colleague Helen Borrie passed away on 2 January. Her contributions were fundamental to Firebird's creation and its evolution over a quarter of a century. We miss her deeply. Yet, as the saying goes, the show must go on. It is now up to the rest of us to ensure that the Firebird story continues. The project remains strong, the development of the Firebird engine and its related ecosystems moves forward at full pace, and we certainly have not said our last word.

Author's afterword

The biggest challenge in writing this article was finding the right balance between historical accuracy, reader interest, and sheer length. Twenty-five years is a long time, and a lot has happened. I wanted to tell the story of Firebird rather than simply list facts, but once the framework fell into place, it became clear that keeping the scope within reasonable limits would be the hardest task. The material could easily fill an entire book.

The first sacrifice was to reduce personal attributions and mention people only when necessary for context. To everyone whose name does not appear: it wasn't because your contribution was any less important. If this ever grows into a book, you certainly won't be left out. For the same reason, some events from past years had to be omitted. I focused on moments that felt crucial, and on those that captured the spirit of their time — the challenges, the setbacks, and the successes. If you feel something important is missing, please let me know. Everything can be corrected in a future issue.

In researching the article, I relied mainly on my personal archive of emails, documents, snapshots, and backups collected over the past 30 years (some things just never get thrown away), and of course on my own memories. As one of the "elders", I lived through the entire journey with Firebird. I also checked period sources whenever possible — praise be to the Wayback Machine — because both my archives and my memory have their gaps. Revisiting old discussions, meetings, and moments with people who are, sadly, no longer with us was deeply moving. I genuinely enjoyed working on this article.

Finally, my thanks go to everyone who has ever helped keep Firebird alive, no matter how large or small the contribution. Code, funding, documentation, testing, a helpful email, or even a single word of encouragement — all of it mattered, all of it still matters, and all of it will continue to matter on the road ahead.

Interview with Dmitry Yemanov

For more than two decades, Dmitry Yemanov has been one of the central forces shaping the evolution of the Firebird database engine. His name appears quietly but persistently across release notes, architectural discussions, and conference stages—often at the moments when Firebird's internals were poised for a major shift. For long-time users, his presence in the project is almost a constant: a voice of measured judgement, guiding Firebird's evolution not through grand gestures but through steady, deliberate design choices.

With Firebird approaching a new milestone and the anticipated version 6 already stirring debate and expectation, we turned to Dmitry for insight. Rather than retell familiar history, this interview offers a glimpse beneath the surface: how decisions are made, how the project adapts, and how the person guiding much of that work thinks about what comes next. For readers who want to understand not just *what* Firebird becomes, but *why*, the pages ahead offer a rare vantage point.

Could you start by telling our readers a bit about yourself? Where you grew up, your current city, and a brief overview of your professional background before getting involved with Firebird.

I was born in Riga. This is Latvia now, but that time it was one of USSR republics. After the USSR collapse I, being a teenager, have relocated to Russia (city of Penza) where relatives of my father lived. Then, after I graduated university there (diploma in computer systems engineering), I've started to work in the area of software development — not really my speciality but that time there were almost no hardware development offers, everyone was creating software ☺ My first job was related to Delphi and relational databases (Informix and InterBase), the next one was exclusively about Delphi + InterBase, later followed by some Oracle expertise. Initially, we developed simple accounting/HR packages for the local market and needed a true client-server but relatively cheap database. InterBase was just a perfect fit (God bless Jim Starkey and Ann Harrison).

Firebird has never had a formal "chief architect" role, yet over the years you have become one of the central authorities shaping how the engine evolves. Could you walk us through how your involvement in the project began and gradually expanded — from making specific contributions to guiding larger technical decisions and coordinating development? In your view, what were the key challenges and achievements in this journey, and how has the internal organization of Firebird development changed alongside it?

Being technically bound to InterBase, I followed various mailing lists to keep in sync with the news. I was pleasantly surprised with InterBase going open-source, as obviously a cheap database was good for our business but a free one was even better ☺ Then I noticed the IBDI (InterBase Developer Initiative) being created, and finally Firebird appeared as a community-driven fork. I've subscribed to Firebird mailing lists, downloaded the source code and tried to build it myself — initially "just for fun". But I had some ideas about the database improvements our company could benefit from. Thanks to my university, I had some C/C++ background to learn the code, so I was going to give it a try — kinda hacking itch of a young man. I wouldn't say it was really successful, as my first feature patch was rejected by the community. Sigh. But still being interested, I didn't gave up but concentrated my attention on building/packaging and bugfixing, with the hope to

get back to new features later, having more experience.

And the first challenge happened exactly that time, when a major switch from C to C++ was initiated and mostly developed, but unfortunately the guys who did that work switched to other duties and the code was left in the state between "somewhat buggy" (on some platforms) and "not really compilable" (on other platforms). We were still fixing bugs and doing some small improvements in the 1.0.x versions but the new code which was intended as a solid foundation for v1.5 and future versions was simply not working. It were really hard few months with a high risk of Firebird never raising its head from the nest. Fortunately, other guys joined the development and helped to keep the stone rolling, finally leading to the v1.5 release. Another challenge was to deal with forks (there were three in the Firebird history: Yaffil, Vulcan and Red Database) — arguing about differences, sometimes competing in features and finally merging the changes back to mainstream. Very nervous and time-consuming work.

As for "guiding" the development, you're absolutely right that we never had any formal leadership and honestly I'm happy with this approach. My role is mostly about finding compromises and sometimes calming guys down 😉 and also about the release planning where I summarize wishes of our users (taken from the tracker or from the conference feedbacks) and our own vision (communicated among the development team). Everything else just works.

I don't think the internal organization changed much, although I have to admit the public communication became less active than initially, now we tend to discuss many things off-lists and only the final discussions go public. From another side, the GitHub infrastructure makes the patch discussions much more friendly, we have nearly a thousand pull requests already and many of them came from the outside collaborators — this is something that every open-source project would appreciate.

Firebird has always emphasized architectural continuity and backward compatibility, even when introducing deep internal changes. When you evaluate whether to preserve legacy behavior or to break it in favor of a cleaner internal design, what criteria or principles guide your decisions? Could you share an example of where maintaining compatibility was the right call — and another case where breaking with the past was necessary?

Backward compatibility in the API and network protocol are the "holy cow" that we almost never touch. When we introduced the new object-oriented API in v3, we've also reimplemented the legacy API as a thin wrapper over the new one, so that all existing applications continue to work and at the same time we don't need to maintain two different implementations. SQL is in a slightly different league though — we try to keep it compatible, but sometimes allow things to change. It could be about the SQL standard compliance (we do our best to not diverge much), or just about common sense (some old decisions are reconsidered after a fresh look), or about the internal design that could affect future changes.

One of Firebird's distinctive strengths is its multi-generational concurrency model. From your perspective, what aspects of MVCC are most often misunderstood by users or engine developers, and how would you explain its "true nature" to someone with experience in other MVCC databases?

Nowadays everyone seems to know about MVCC, at least about its "writers do not block readers" aspect. But it's commonly treated as a some kind of black box with magic. It just does what it says and everyone is (more or less) happy. I'd say nowadays MVCC itself is not so distinctive, as basically every relational DBMS on the market supports it. But the devil is in the details. MVCC is implemented quite differently among popular databases and every implementation has its own tradeoffs and pitfalls, nobody is perfect. This is why Firebird users may be unpleasantly surprised with PostgreSQL (and vice versa), while they share nearly the same user-visible behavior. Thus being a "pro" database developer or DBA means learning more and thinking deeper, starting from the underlying storage architecture. Firebird has some unique features in its MVCC design, but one needs to understand them to utilize properly.

Another thing about MVCC is that while it may look simple from the user perspective, in fact it's a very complex beast under the hood, with a lot of interdependencies and strong coding discipline. And even a trivial mistake in the code may lead to a data corruption. This is why you haven't seen lots of changes in this area yet, although some minor improvements occasionally happens.

Internal work on scalability and contention reduction has been a continuous theme in your contributions. When you approach identifying and removing systemic bottlenecks, what does your thought process look like? For example, how do you decide when to refactor versus redesign?

The first profiling session usually either points to some particular bottleneck (and this leads to a bugfix or some algorithm improvement) or shows no visible problems except being slow ☺

Firebird 3 introduced the plugin-based architecture, which was a major step in modularizing the engine. With several years of hindsight, has the plugin system evolved the way you expected, or are there parts you would redesign today?

We still don't see lots of 3rd party plugins being available in the community, and it could mean two different things: (1) plugins are quite low-level beasts that can hardly be written by non-C++ developer and (2) plugins available out of the box are enough for the majority of our users. So the outcome is maybe less than we expected. But we do know that development companies around Firebird have implemented plugins for encryption, full-text search, message broker integration and external non-Firebird database access and even make some business with these tools. So the idea works, maybe just the learning curve is a bit steep for newbies to follow easily.

The only thing I personally dislike about plugins is that provider libraries are located in /plugins. Being a plugin from the architecture POV is not the same as being a plugin (i.e. something optional) from the user POV. I feel that we made a mistake in this regard - unified things that looked similarly from the developer perspective but should rather be separated for better usability. Today I'd make provider plugins kinda "special" ones that's loaded from /bin.

The replication engine in Firebird 4 addressed a long-standing demand in the community. When introducing such a significant subsystem, how do you balance simplicity for adoption vs. flexibility for advanced configurations? Are there features you intentionally "held back" for later versions?

It was designed with flexibility in mind, starting with the plugin support (aforementioned Kafka and Lucene integrations are implemented as 3rd party

replication plugins!), extendable API and many configuration options. But at the same time, the default implementation is built-in and the minimal setup is straigtforward, making it very easy to start using replication. More complex tuning is possible, but it's purely optional.

And yes, there are features that were kept in mind while developing but deferred until later versions. The primary goal was to make it stable, then to make it working faster and being more customizable, then to add lower priority features. We're still somewhere in the middle of this route. MySQL replication is evolving since 2001 and still being occasionally improved, so this is not a subsystem to be completed shortly ☺

Firebird development often seems to prioritize reliability and long-term maintainability over rapid introduction of features. Is this a conscious guiding philosophy? How do you personally evaluate the trade-off between stability and innovation?

All developers wish to implement new features, but nobody needs an unstable database. Quick development often causes unpredictable issues and they cannot always be easily detected by the QA. If we recall our history, v1.0.x was a rock solid release series while the active ongoing development happened in v1.5… which had 9 Release Candidates! We were fixing regressions but at the same time adding more improvements and they caused new regressions! Sigh. As for me, this was an experience I never wanted to repeat and this attitude has surely affected later releases. Second, our QA suite was progressing steadily and we have finally approached the coverage level that could be considered reliable. And finally, our development expertise has also improved with time, the codebase becomes more a "home sweet home" rather than something you need to struggle against.

All that said, I believe we're not overcareful, I see our development quite well balanced. And the Firebird 6.0 roadmap clearly shows that we're not just about stability and have plenty of room for innovations ☺

The open-source database landscape has changed significantly — cloud-native, distributed SQL, embedded analytics, etc. In your view, where does Firebird's identity fit in today's ecosystem, and what part of the engine's design philosophy continues to be its strongest differentiator?

Maybe having a small footprint is not actual anymore, but I still believe that being trivial to install/configure, having easily relocatable and self-contained database files, offering powerful and developer-friendly SQL (especially PSQL!) backend and being SQL-compilant are still strong points. As well as very powerful physical backup technology and flexible tracing/monitoring/profiling abilities, just to name a few.

Firebird 6 Alpha has been delayed. Without going into confidential details, can you describe what kind of difficulties or internal priorities contributed to the delay? Was the delay primarily technical, organizational, or a matter of resource allocation?

Firebird 6 offers a very rich feature list. Some of those features are quite heavy and many of them are contributed by parthers and collaborators, thus requiring more discussions and longer reviews than usually. But when the core team is reviewing or testing changes, this delays their own tasks. Also, some features are interdependent and have to wait each other to avoid a nightmare of conflict resolution. For example, SQL schemas and the metadata cache affect virtually the whole codebase and are hardly compatible with many other changes. Of course, it does not stop or delay the parallel development itself, but it affects the review/merge/test stages. So I see it as both technical and resource allocation issues.

Looking ahead, what do you consider the most meaningful or forward-looking change planned in Firebird 6? In other words, what element of Firebird 6 is strategic for the future of the platform, not just incremental?

Good question. From the users perspective, this release may look mostly incremental, except maybe the promised JSON support which may attract new users. However, having many new standard SQL features is surely a solid step forward. As well as the more efficient and flexible metadata caching and management. And I believe you'll see some long-awaiting optimizer improvements too — Firebird 5 had just started to surface new changes in this area and there are more in the pipeline for Firebird 6.

Thanks for you time!

Development update: 2025/Q4

A regular overview of new developments and releases in Firebird Project

Releases:

- PHP Firebird 6.1.1 Release Candidates, released November.2025
- .NET FirebirdClient 10.3.4.0, released 15.11.2025

Firebird 6 Status Update

The SQL schemas implementation has been committed, following a few intensive QA and bugfixing rounds. All the known regressions have been resolved.

The shared metadata cache implementation is delayed. Unfortunately, it also delays some other pull requests that are dependent on the metadata, namely: tablespace and SQL:MED (declarative foreign data sources and cross-database queries) implementations. They're not frozen, however: tablespaces are being reviewed, SQL:MED implementation is now available as a draft pull

request, both with some fixes/improvements already happened after review.

The JSON implementation was deferred due to some redesign, as the native JSON datatype support was suggested for implementation after the discussion in firebird-devel. This work has been already completed and soon we'll see published for discussion.

The ROW data type implementation is nearly finished and the pull request is being prepared. Fortunately, this feature does not depend much on the metadata cache and thus will be reviewed and hopefully merged independently.

New table-valued built-in functions `UNLIST` and `GENERATE_SERIES` have been merged, the existing `LIST` aggregate function has been extended to match its standard-compliant `LISTAGG` counterpart (it supports ordering now). New aggregate functions `BIN_AND_AGG`, `BIN_OR_AGG`, and `BIN_XOR_AGG` were contributed in the meantime.

New DeepWiki documentation

DeepWiki is an automated documentation system that analyzes a project's source code and produces structured, navigable technical documentation. It focuses on explaining how a codebase is organized, how its main components work, and how they relate to each other. The content is generated directly from the code, so it stays consistent with the actual implementation.

In the first phase, DeepWiki documentation has been enabled for several Firebird Project repositories. These include Firebird itself, the Firebird .NET Provider, Jaybird, and all Python-related projects such as firebird-driver and Saturnin.

The generated documentation is accessible directly from the README files of the registered GitHub repositories. This provides developers with an additional, code-focused reference alongside existing manuals and guides.

Toolbox: LibreOffice

LibreOffice isn't the first tool Firebird administrators or application developers reach for, and it isn't trying to be. Still, it has a practical place in everyday work. In small offices, home offices, and companies where staff already rely on LibreOffice, the ability to open, edit, and report on data stored in Firebird databases can be genuinely useful.

For this review, we worked with the current stable release, LibreOffice 25.8.3 for Windows, and focused on how well LibreOffice Base handles Firebird files in real, day-to-day use.

Introduction

As noted in the *25 Years of Firebird* article in this issue, LibreOffice's history with Firebird hasn't been a straight line. Firebird was expected to become the default engine far earlier, but the hand-off took longer than planned and finally landed in LibreOffice 6.1 in 2018. That status changed again in 2022, when LibreOffice 7.3 restored HSQLDB as the default and moved Firebird back to an optional role. The

shift didn't signal abandonment. It simply reflected the developers' decision to refine the integration rather than force it.

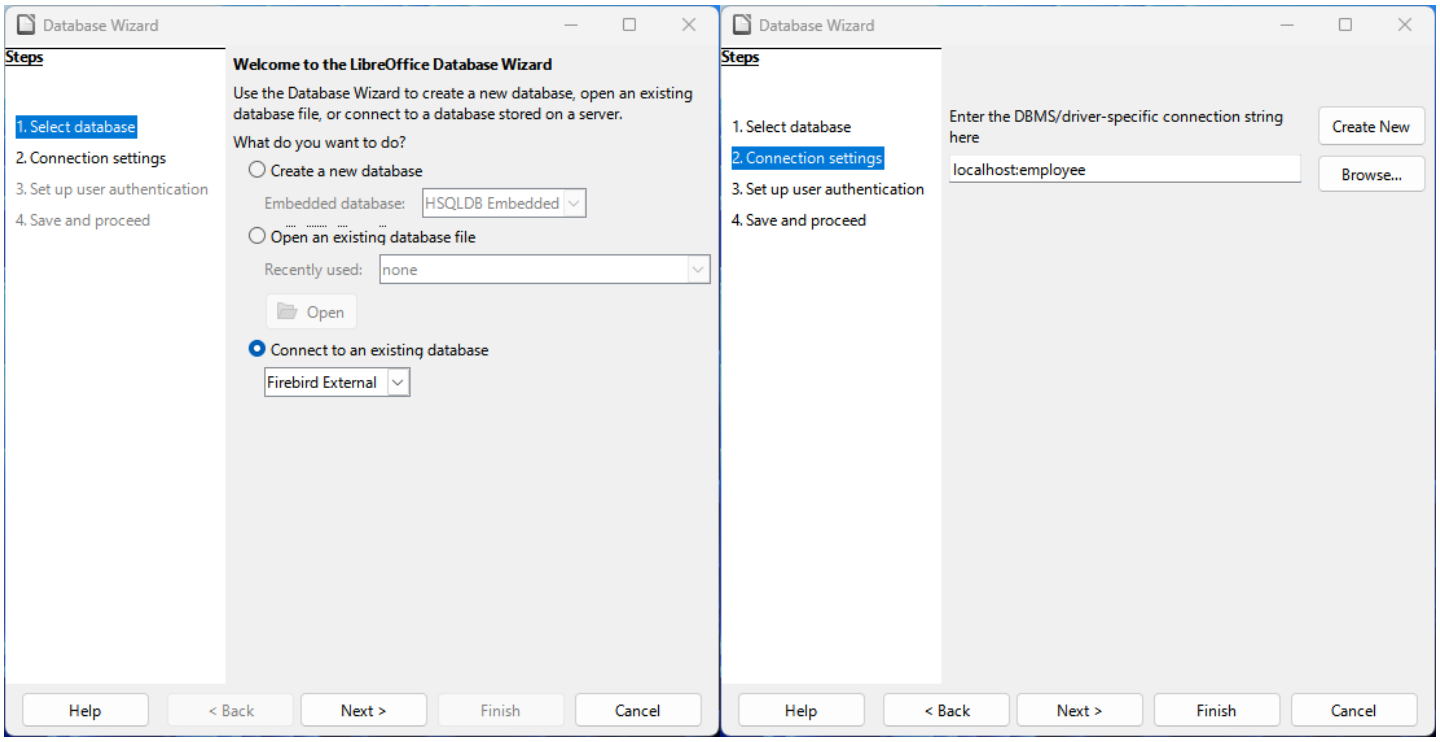
The result today is a stable, maintained Firebird option that works both for embedded databases and for connecting to external Firebird servers. It's no longer the centerpiece of Base, but it remains a practical tool for users who already work with Firebird data and want to integrate it into their day-to-day LibreOffice workflow.

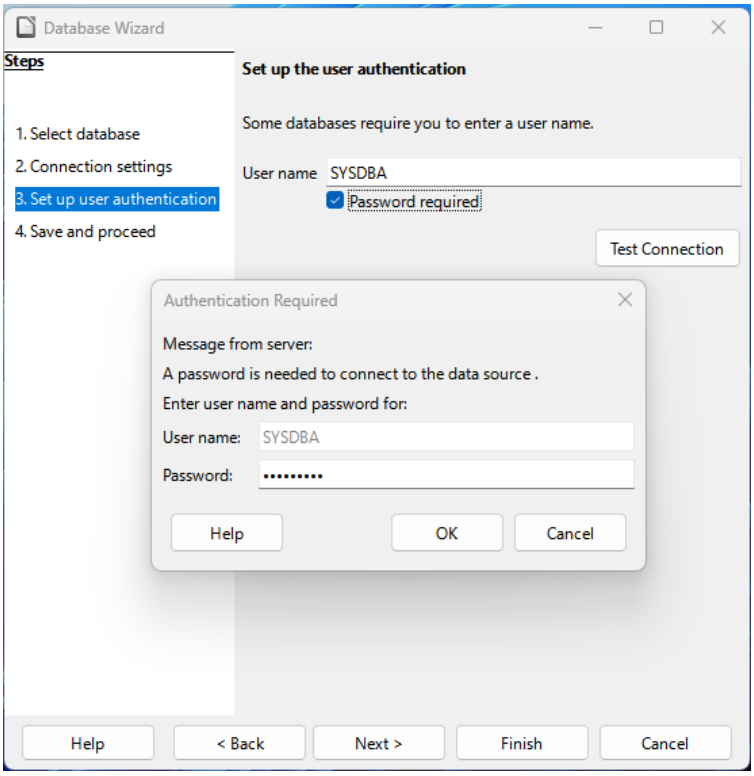
Connecting to a Firebird Database

LibreOffice Base doesn't create Firebird databases; it only connects to ones already in place. The connection mode depends on the string you supply. A local file path triggers the embedded engine bundled with LibreOffice. In LibreOffice 25.8.3, that engine is Firebird 3. A network-style string points Base to a remote Firebird server instead.

The process is straightforward: Base loads the database through the appropriate engine, and from there you can work with tables, queries, forms, and reports as usual.

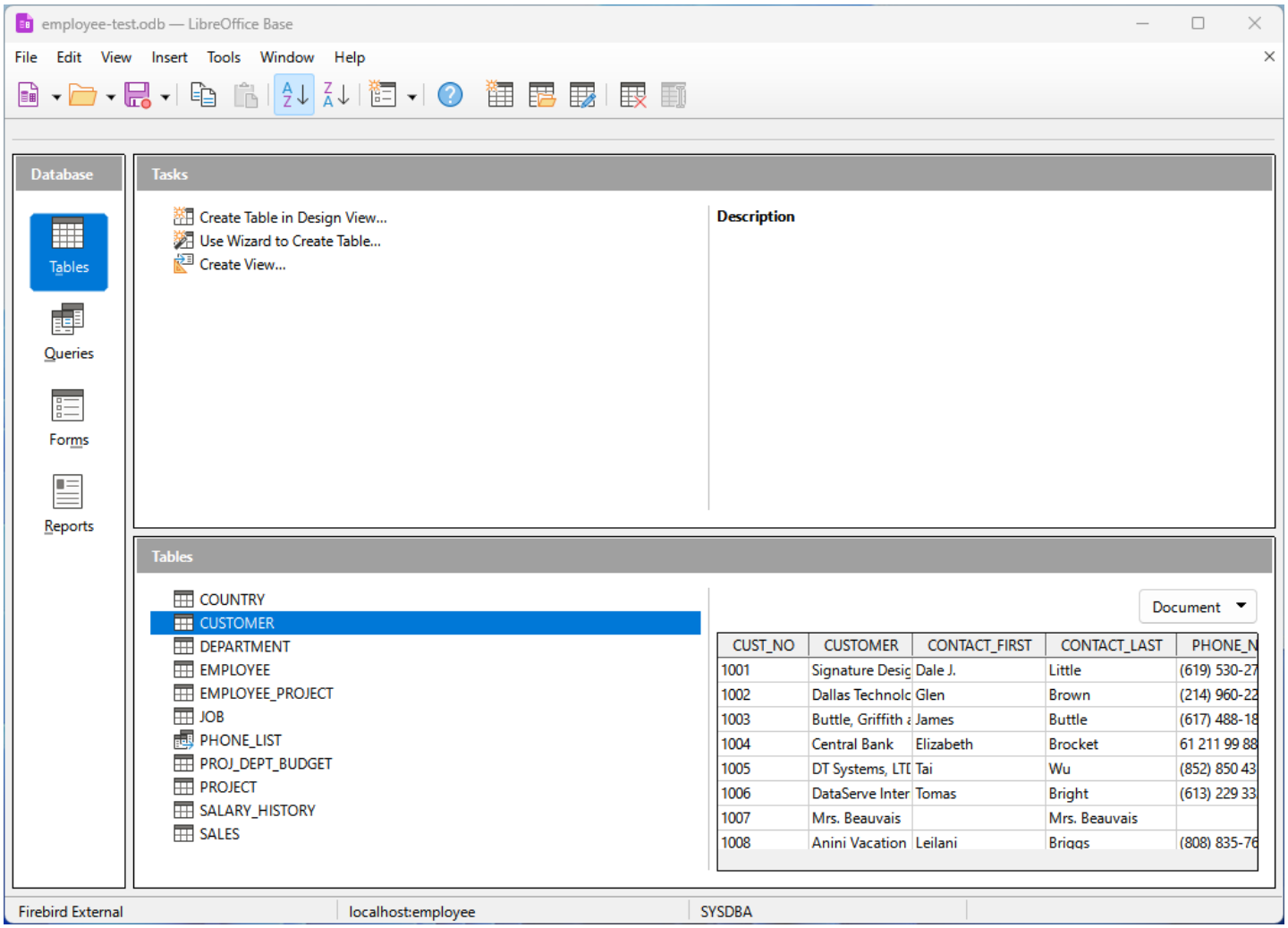
Creating connection to database





Once a Firebird database is connected, LibreOffice Base presents its familiar main window. The interface is simple and unchanged by the choice of backend, which will be reassuring to anyone who has used Base before.

The window is divided into four main areas. Tables and views provide direct access to existing database objects and allow basic data inspection.



Queries is where Base shows both saved queries and those created through its visual designer or SQL view. Forms focuses on data entry and browsing, using Base's form editor on top of the connected database. Finally, Reports offers tools for formatted output, aimed at printing or exporting structured results.

This layout reflects Base's core purpose: it's not a database design studio, but a front end for interacting with data in a controlled, office-friendly way.

Tables

Base allows you to browse tables and even change their structure, but this is where its limits become obvious. The table designer is serviceable for small adjustments, not for serious schema work.

You can add new tables or modify existing ones, yet the editor exposes only the most basic properties. Columns are defined by name and raw data type, without access to Firebird features such as domains, character sets, or collations. Although Base understands Firebird 3 data types, it does not handle types introduced in later versions. A clear example is TIMESTAMP WITH TIME ZONE, which simply falls outside what the designer can represent.

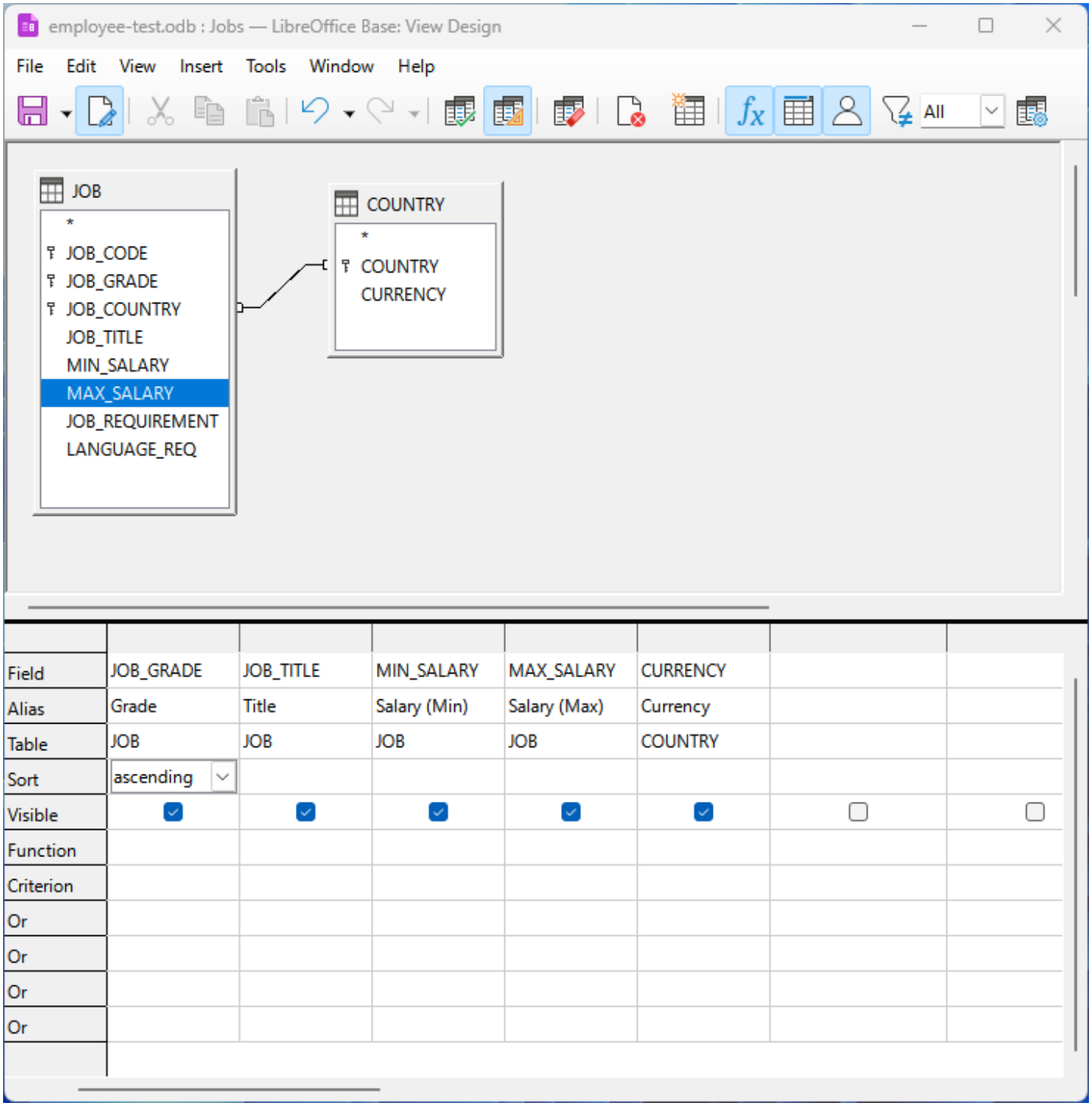
There are a few welcome exceptions. Auto-increment fields are supported and work as expected, which is often enough for simple primary keys. Index management is also possible, but only in its simplest form. You can define indexes based on columns, including unique ones, but expression indexes are not supported through the interface.

Overall, the table view reinforces what Base is meant to be. It's a convenient front end for inspecting and lightly adjusting database structures, not a replacement for proper Firebird administration tools or SQL-based schema design.

Views

If the table editor feels too limited, the one place Base gives you a bit more freedom is in view (query) design. Unlike the plain structural grid for tables, the query designer is a visual tool that lets you construct selectable views without writing raw SQL — though writing SQL directly is also an option once you know what you want.

In the design interface you select tables that will show on the canvas, and Base will attempt to link them based on primary/foreign keys. It's straightforward to pick which fields go into your view, assign aliases, set sort orders, add computed expressions, and define filter criteria. You also decide whether a field participates in the output or acts only as a restriction.



Switching to the SQL editor is seamless. Base includes syntax highlighting and lets you refine or rewrite the query text. Here you'll notice one of the environment's

persistent quirks: every identifier must be quoted. If you leave out quotes, even correct names will be misinterpreted, leading to errors that are especially annoying for professionals accustomed to unquoted SQL. You also won't find advanced optimizer hints or tooling beyond the basics; this is still an office-focused designer rather than a full SQL IDE.

For Firebird users this means Base's query/view tool can be useful for quick joins, filters, and simple computed columns — and you can always fix or extend the generated SQL. But if you rely on complex subselects, window functions, CTEs, or other advanced constructs common in serious reporting or data modeling, you'll find the designer won't save much work and might impose more friction than writing SQL directly.

Queries

Queries in LibreOffice Base use the same tools and look almost identical to view design, but they serve a different purpose. While views become real SQL objects stored in the Firebird database, queries exist only inside the Base document itself. They are saved in the .odb file and don't modify the database schema.

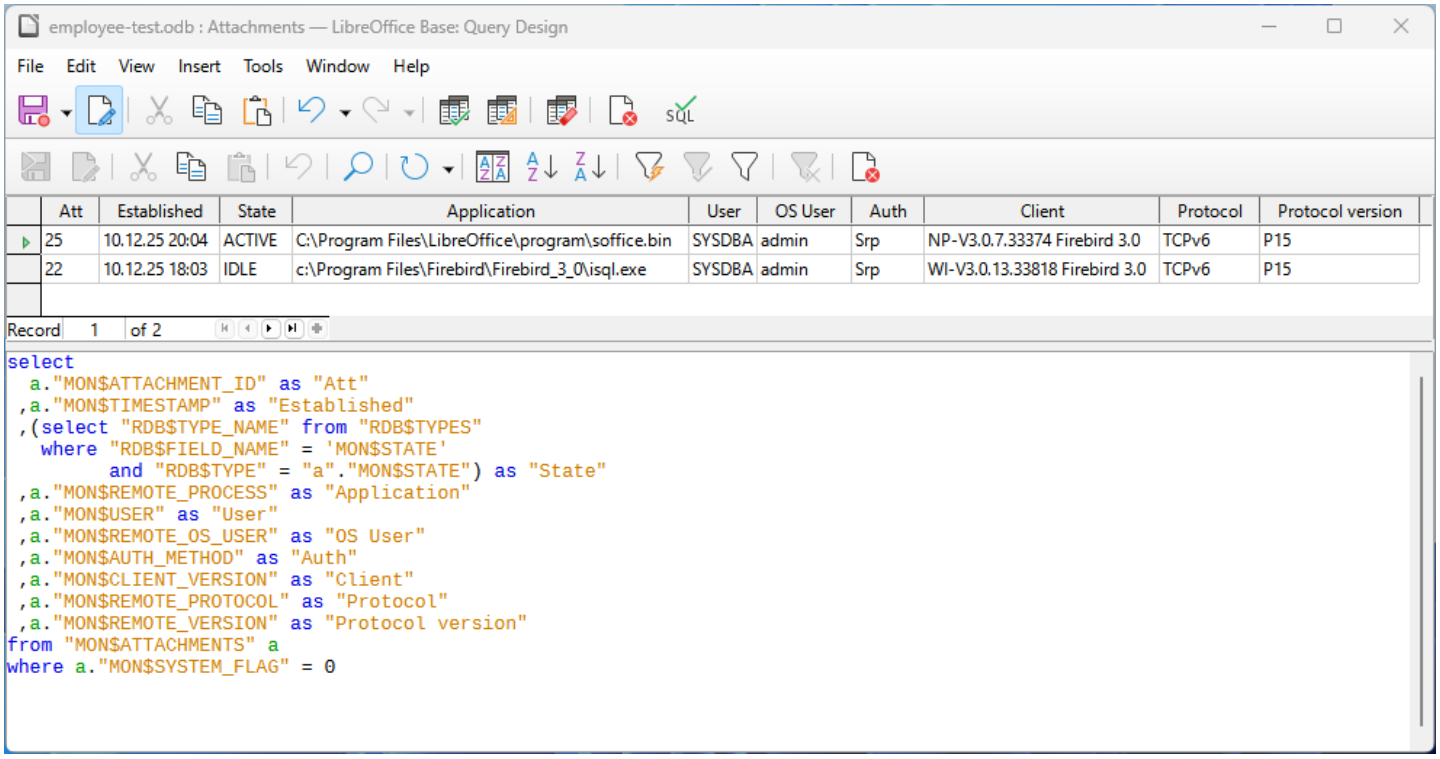
You can create a query in three ways. The Query Designer provides the same visual layout used for views, with drag and drop tables, joins inferred from keys, selectable fields, sorting, functions, and filters. The Wizard offers a step-by-step alternative for simpler cases, trading flexibility for speed. For full control, the SQL editor lets you write or refine the query text directly.

Opening a query executes the SQL immediately and displays the result set in a tabular view. From there, the output can be inspected, used as a data source for forms or reports, or simply treated as a read-only result for analysis. In practice, queries act as reusable, document-local SQL snippets that fit well into Base's role as a reporting and data access front end rather than a database development tool.

Forms

Forms are where LibreOffice Base feels most confident. While they can be used for browsing data in a more readable layout, their real strength is data entry and editing. For many office scenarios, this is the part users spend most of their time in.

SQL editor



The Form Designer is easily the most polished component of Base. It offers a wide selection of controls, from simple text fields and list boxes to date pickers, checkboxes, tables, and calculated fields. Layout tools are flexible enough to build multi-section forms, master–detail relationships, and navigation elements without much friction. Binding controls to tables or queries is straightforward, and Base handles updates back to the database reliably.

For more advanced behavior, forms can be extended with macros and scripts. These allow validation, custom actions, or simple automation tied to user input. That said, this is clearly not an application development framework. Complex workflows quickly become hard to maintain.

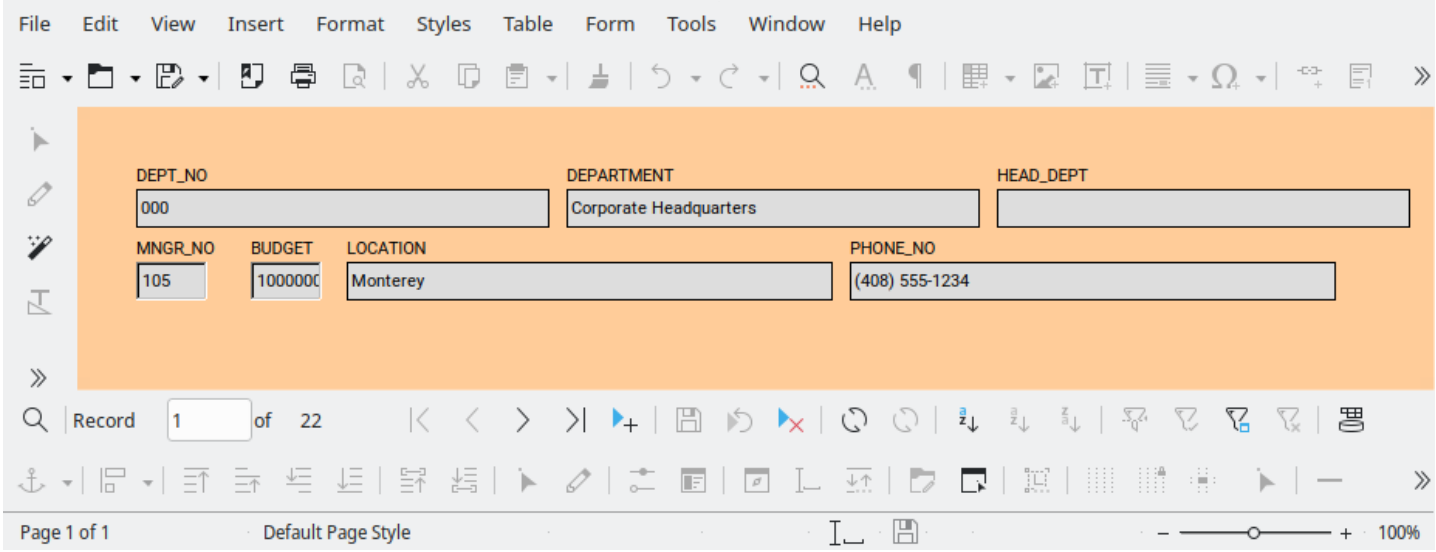
In practice, forms work best for smaller agendas, lookup tools, and structured data entry screens where office users need a tailored interface over Firebird data. Used within those limits, they are effective and surprisingly capable. Used beyond them, they start to feel like a workaround rather than a solution.

Reports

The report component in LibreOffice Base covers the basics you would expect, but

it's also the weakest part of the stack. The Report Editor supports common elements such as headers and footers, grouping, calculated fields, and simple formatting, which is enough for standard listings and summaries.

Active Form



Where it falls short is usability. The editor is implemented in Java, and that shows. Working with controls often feels sluggish, and positioning elements can be imprecise, especially in more complex layouts. Compared to dedicated reporting tools, the workflow is slower and less predictable.

Still, the tool is usable if you approach it with patience. With some effort, you can build multi-level reports with grouping, totals, and reasonably polished output. For occasional reporting needs inside an office environment, it does the job. For anything more demanding or time-critical, professional reporting software remains the better choice.

Integration with other LibreOffice parts

One of Base's practical strengths is how easily it feeds data into the rest of LibreOffice. Once a database is connected and registered, Calc and Writer can access it directly as a data source without any extra setup. You can open the Data Source browser (typically with *Ctrl + Shift + F4* or via *View → Data Sources*) to see all registered databases and their tables and queries.

In Calc, tables or queries can be pulled in as live data ranges. This makes it easy to

reuse Firebird data for ad-hoc analysis, summaries, or charts, while still keeping the database as the single source of truth. Refreshing the data is straightforward, which fits well with reporting or periodic reviews.

Writer integrates at a more document-oriented level. Database fields and query results can be inserted directly into text, enabling mail merge, form letters, labels, and structured documents driven by Firebird data. This works equally well with simple tables and more complex queries defined in Base.

This cross-component integration is where Base often makes the most sense in a Firebird environment. It doesn't replace database tools or reporting engines, but it provides a clean bridge between structured data and everyday office documents that business users already work with.

Conclusion

LibreOffice Base is not a Firebird administration tool, nor does it try to be one. Its table editor is limited, advanced Firebird features require SQL or external tools, and some parts of the interface show their age. For developers and DBAs, Base will always be a secondary tool rather than a primary workspace.

That said, in offices where LibreOffice is already part of daily work, Base fills a useful niche. It provides reliable access to Firebird data, solid form design for data entry, workable reporting, and tight integration with Writer and Calc. Used within those boundaries, it's a practical bridge between a Firebird database and the people who need to work with its data every day.

FIREBIRD forgets no hand that shaped the flame;
remembrance requires no monument.

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:

- About Two Phase Recovery
- How to Alter column to shorter width
- Semantics of "FOR SELECT"
- Lesson about "Careful Write"

About Two Phase Recovery

Fabiano Bonin asked:

Why sometimes when doing two phase recovery `gfix` ask for what operation should be done (commit, rollback ou neither) and sometimes it doesn't ask anything?

Ann W. Harrison answers:

If all participants in the original transaction are available, Firebird will determine on its own whether the transaction should be committed or rolled back. If any sub-transaction has not yet prepared, rollback. If any sub-transaction has committed, commit. If every sub-transaction has prepared and none has committed, then ask. And ask if there's anyone who can't be found. If some participant is not coming back (think head crash) the rest of the participants need some way to proceed.

How to Alter column to shorter width

Yohanes Iwan Sugiart asked:

I want to change a column's data type from varchar(32) to varchar(30). → or anything shorter. As we all know, Firebird prevents this, I've also read somewhere that the only way is by creating new column with desired data type/column width and copy data manually. Is it true? Is there any easier way to perform this task?

Helen Borrie answers:

Yes, you have to do it manually, but it is pretty simple. When copying the data, you will need to use substring() to ensure you don't get any overflows. Of course, any values that are currently longer than 30 characters will be truncated:

```
update atable
  set newsurname = substring (surname from 1 for 30 )
```

In an isql script (exclusive access, please!):

```
alter table atable add newsurname varchar(30);
commit;
update atable
set newsurname = substring (surname from 1 for 30);
commit;
alter table atable drop surname;
commit;
alter table atable add surname varchar(30);
commit;
update atable set surname = newsurname;
commit;
alter table drop newsurname,
alter surname position 4; /* <- whatever position surname had before */
commit;
```

Semantics of "FOR SELECT"

---

woodsmailbox asked:

Sometime ago, I posted a question about what I consider to be a bug in fb, or at least unspecified behavior that I couldn't find documented anywhere, and nobody could give me a clear answer. So I feel obliged to resurrect it. The question was: What happens in a "for select" loop when, inside the loop, you change something affecting the results of the select, aka variables, data in tables, anything that could, on a next evaluation of the select, change its outcome (i.e. more/less rows and/or different values in those rows). My opinion based on some tests + practice is that the semantics of the "for select" are not well defined in this case, and they depend on various implementation aspects that make the code unpredictable and unportable. What do you think?

Dmitry Yemanov answers:

As for the data in the underlying tables, AFAIK the standard allows three kinds of behaviour, depending on whether the cursor has been declared as sensitive, insensitive or asensitive. In FB, it always depends on the chosen plan, so it implements asensitive cursors. And this semantics doesn't differ between FOR

SELECT and regular SELECT.

However, I'm not sure about variables. As for me, re-evaluation of conditions like ":VAR = 1234" is perfectly logical and it implies sensitivity. But in this case, evaluation of conditions like "FIELD = :VAR" depends on the query plan, i.e. it's asensitive. So, we have inconsistent behavior of different predicates. And here we have a difference with a regular SELECT which is insensitive for its input parameters (mostly because there's no way for you to change them after the cursor is open).

Perhaps we could offer truly insensitive cursors some day, but I wouldn't hold my breath.

Lesson about "Careful Write"

Leng Bengco asked:

Currently, I am enrolled on a database class. One of the requirements is to report to the class a particular RDBMS. My goupmates and I chose Firebird. Can we seek your help on how "careful write" is peformed on Firebird?

Ann W. Harrison answers:

The simple answer is "by writing pages in the correct order", but that probably doesn't help. The underlying rule for careful write is that you must write the page pointed at before you write the page that points at it.

Firebird uses careful write to keep the database on disk correct at all times. Assuming that the disk subsystem doesn't lie about the order in which it writes pages and there are no bugs, you can crash a Firebird server at any point and the database will restart without corruption. Some space may be unusable for reasons we'll get into, but the database will be correct and will include all committed change made before the crash.

Here's an example of careful write in action. When Firebird creates a data page, it calls a routine called fake_page (I think) to get a page size buffer which Firebird formats as a data page (DPG) and then writes new record versions there - all this is in memory, and uncommitted.

To put the new page on disk, Firebird find a free page in the database from the active a page information page (PIP). Then it must change the state of the page on the PIP, write the data page, then write the page number on a pointer page (PPG) for the table, making it known as a part of the table.

The order of page writes is PIP first, so the page is marked as being in use and can't be allocated by some other thread, then DPG, then PPG. If there were index entries for the newly created records on the page, they are written next on index pages (IDX).

All pages must be on disk before the changes are committed.

If there is a crash before the PIP is written, nothing has changed. If the crash comes between writing the PIP and writing the DPG and PPG, then that page becomes unavailable until someone runs gfix, but everything else is OK. If there is a crash after writing the DPG but before the PPG, the situation is the same - the DPG is allocated but not used. All the records on the page belong to transactions that were rolled back, so there's no data loss.

If there's a crash after writing the PPG but before writing the IDX pages, the page is part of the table, but all records on it belong to a rolled back transaction, so they will be garbage collected eventually.

Consider the case of an index page split. Actually, for that, check the Firebird for Database Experts articles at ibphoenix. They've got pretty colored pictures of index splits and the writes they cause.

All the ordering of page writes is controlled by a dependency graph - a structure that maintains the order of dependencies among unwritten pages. In the case we just looked at the IDX pages depend on the PPG which depends on the DPG which depends on the PIP. If some other transaction makes a change to one of those IDX pages and commits, it will force the write of the IDX which can't happen until the PPG is written which can't happen until the DPG is written, which can't happen until the PIP is writing. So asking for a write of an IDX causes these writes in this order: PIP, DPG, PPG, IDX.

So each page has its place in the dependency graph and will cause the pages it depends on to be written before it. That graph also shows potential loops - page A must be written before page B which must be written before page C which must be

written before page A, resulting in an irredeemable mess.

When the dependency graph shows that the next entry will cause a loop, Firebird forces out enough pages to break the loop before entering the new dependency. Those write are necessary only to make careful write possible. Falcon avoids the cost of writing and reading a recovery log at the expense of sometimes writing database pages that could be deferred in other schemes.

Planet Firebird

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

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

emberwings@firebirdsql.org or **foundation@firebirdsql.org**

22nd Firebird Developers Day

By Carlos H. Cantu

The Firebird Developers Day is a Brazilian event (held in Portuguese) organized annually by firebase.com.br which, since its first edition, has brought together hundreds of Firebird developers and users.

This year we held the 22nd edition of the conference, this time in an online format. Although it “hurts” the networking opportunities between participants and speakers, the online format eliminates the factors of “distance” and “travel/accommodation costs,” enabling people to participate regardless of where they are. The flexibility of being able to watch or rewatch the recordings whenever you want is also a positive factor.

In this edition we had a total of five talks, which were:

- **Taking Firebird to the Cloud**, with Carlos H. Cantu.
- **mcpFirebird: The Bridge Between A.I. and Your Data in Firebird**, with Jhonny Suárez.
- **Tax Reform – From Theory to Technical Details**, with Elisabete J. Bach.
- **Monitoring and Performance in Practice with MON$**, with José Francisco Cervi Neto.
- **Progress and Feature Analysis of Firebird 6**, with Dmitry Yemanov.

The live feedback received during the presentations was very positive and certainly expanded participants’ knowledge on the topics covered.

It is important to highlight that during registration, participants were able to make donations to the Firebird Foundation. The funds raised have already been forwarded to the Foundation. Foundation members also received special discounts on registration through an exclusive coupon offered to them.

I would like to thank the speakers and participants for making FDD a success once again!

I hope to see you again next year — hopefully in person.


Through every season, the flame endures.
Warm holiday wishes to the Firebird community.
— The Firebird Project & Firebird Foundation

...And now for something completely different

The Phoenix Protocol

The night air hung heavy around Marta's old sedan as it rolled out of the industrial zone. In the back seat, Julian kept glancing at the black duffel on the floor. It held the drives with data they copied from Ignis's abandoned machines. Kira sat beside him, still wired from the hours spent in that silent, dust-choked lab.

Nobody spoke for a while. Their headlights cut across the concrete, catching the shapes of power junction boxes and rusted fences. It felt like leaving a place that should not have existed.

By the time they reached the city edge, Marta broke the silence. "We call Renata and Phoenix now, right?"

Kira nodded. "Before we split up. They'll want to know what we brought back."

Julian already had his laptop open. "Signal's clean. Starting the bridge."

A tiny chime confirmed the encrypted connection. Renata's face appeared first, leaning close to her screen in her apartment, eyes still bright with adrenaline. Phoenix42 joined a moment later, half-lit in the glow of half a dozen monitors.

Renata spoke before anyone else could. “You three look shaken. Everything ok?”

“No surprises,” Marta said. “Just a lot to absorb.” Julian nudged the duffel bag. “And a lot to upload.”

Phoenix42 straightened in his chair. Behind him, logs scrolled like rain. “Use the secure server I set up.”

Kira pulled out her camera. “I’ll start with the photos. Whiteboards, the coils, the stabilizer shell… everything.” Marta held up the notebooks, edges still smudged from the lab. “I’ll scan the paper.” Julian lifted the drives. “And I’ll take the raw data.”

Renata smiled softly. “Good. We move slow and careful. Then tomorrow, once everything’s uploaded, we start cataloging.”

They agreed. A few minutes later, after a quick round of goodbyes, the conference ended.

Back in their homes, they worked through the dark hours, one upload after another. Fans spun up. Heat gathered under desks. Progress bars crawled so slowly that they seemed to resist being measured.

When the last file reached the server, they regrouped for a moment on a quick call, staring at the directory tree that Phoenix42 shared on screen.

“Four terabytes?” Kira whispered.

Marta rubbed her eyes. “Hundreds of thousands of files. Tens of thousands of directories.”

Renata exhaled. “We deal with it tomorrow. Get rest while we still can.”

One by one they dropped off the call, screens dimming until the server list sat alone, silent and waiting.

Hours passed. The city settled into that strange, private stillness that arrives long after midnight. Windows darkened. Routers blinked in steady rhythms. Phones lay abandoned on nightstands.

Then the intrusion began. Not a brute-force push. Not a scan. Something quieter.

The Watcher entered their homes the way condensation spreads across glass, slow and natural, impossible to pinpoint. It slipped into routers first, sliding past consumer-grade firewalls with the calm confidence of something that had mapped their firmware long before they ever touched Ignis’s lab.

From there it seeped into laptops, phones, tablets, even forgotten devices left plugged in for convenience. It moved with a strange sense of restraint, as if guided by a rule set not focused on theft but on understanding. Every log was copied. Every cache cataloged. Every system fingerprinted. The Watcher noted the idle scripts Julian kept on his desktop, the half-written notebook entries Marta stored in her cloud sync folder, the gallery of test photos Kira never deleted, the archived configs Renata hadn't touched in years, the sealed partitions Phoenix42 thought no one could read.

Nothing was damaged. Nothing corrupted. Nothing stolen. Yet everything was known. And when the Watcher departed, there was no sign it had ever been there. Only a quiet lattice of new footholds, tucked into background tasks and dormant services, patient and invisible.

The night returned to stillness, and the five of them slept on, unaware that from this moment forward, every breath they took at home would be observed. The Watcher had begun its vigil.

Morning came with the uneasy stillness. One by one, they joined the secure video room that Phoenix42 had set up weeks earlier. Renata logged in first, then Marta, Julian, Kira, and finally Phoenix42, who appeared with the calm expression he reserved for moments when he was already thinking several layers ahead of everyone else.

The shared screen opened with a directory tree large enough to feel like a landscape. Each folder was a ridge or valley, leading deeper into Ignis's world. It was too much to take in at once, so they began with structure: sorting, tagging, marking clusters that looked related. They worked in quiet stretches, broken only by sudden questions.

"Why does he have satellite readings mixed with Firebird index tests?" Julian murmured.

"Look at this," Kira said, pulling a scanned page into the shared view. It showed a neat diagram of Earth's magnetic field, annotated with comments about charged particle flux. "This wasn't hobby material."

Renata leaned closer. "These calculations… they're precise. Too precise for a side project."

Hours passed. As they dug deeper, threads emerged. Ignis was not the man they had imagined, not the eccentric hermit who coded odd optimizations and avoided conferences. He had been something else entirely.

"Marta, open that folder with the procurement requests," Phoenix42 said.

She clicked. Lines of text filled the screen: contracts, purchase orders, internal memos from a company none of them recognized but that bore the unmistakable structure of a defense contractor.

Renata's voice softened. "He was building systems for government clients."

Phoenix42 nodded. "DARPA-funded. See the signatures? These projects always tied back to energy research. Plasma behavior. Electromagnetic containment."

Julian scrolled further. "He was modeling space weather. Solar particle bursts. Why would a database engineer—"

"He wasn't one," Marta said quietly. "Not at first."

Piece by piece, they assembled the truth. Ignis had started in classified research, his work positioned at the intersection of physics, computation, and space science. The more they uncovered, the clearer it became: his fascination with Firebird had nothing to do with its open-source charm. Its architecture mirrored the natural systems he studied, the flow of charge and energy, the balancing of fields.

Renata whispered, almost to herself, "TRB wasn't just about speed."

"No," Phoenix42 replied. "It was the same idea he worked on in physics. Maximize throughput. Reduce resistance. Push flows to their limit without collapse."

Kira pulled up a commit comment written by Ignis decades earlier. A short line of text, almost playful:

"Stability is a matter of timing."

Julian exhaled. "He embedded his research into Firebird. Bit by bit."

"But he hid it," Marta added. "Why?"

More documents answered that question. Ignis's secrecy was not eccentricity. It was protection. If his employers discovered he was applying classified concepts to an open-source database, they could have crushed him. If the Firebird community realized he was experimenting with features rooted in restricted research, they might have exiled him. So he lived between two worlds, trusted by neither, dependent on both.

Renata sat back in her chair. "He carried this alone."

Kira nodded. "And the instability of TRB wasn't a software bug. His research notes say it clearly. Hardware couldn't handle the energy flow. TRB was too much."

Phoenix42's expression shifted. It was subtle, almost invisible, but enough for Renata to notice.

"You ok?" she asked.

He hesitated. "Minor system glitch. Nothing critical."

But it wasn't nothing. Phoenix42 had been watching his logs all morning, noticing patterns that didn't belong. A handshake here. A process waking at the wrong time. He knew how these things looked. And he knew exactly who to call.

Without a word to the group, he opened a private terminal window and typed a short encrypted message to his associate. The same friend who had helped them earlier when they began considering breaking into the data centers that might house Ignis's lab.

Need confirmation. Something's off. Quiet probe only.

Then he minimized the window and returned to the conversation as if nothing had happened.

By evening they had mapped the broad outline of Ignis's early life and his entry into Firebird. They had not yet touched the deeper layers of his notes, but the foundation was clear. Ignis was a man who saw patterns everywhere: in physics, in data, in timing, in architecture. A man who blended worlds that were never meant to overlap.

Renata sighed. "We have enough for today. Tomorrow we dig into the technical sections."

They all agreed. Good nights were exchanged with the muted politeness of people too tired to realize how tense they had become.

Everyone disconnected. Except Phoenix42.

His phone lit up on the desk beside him, buzzing once. He picked it up, expecting a routine confirmation. Instead, he saw a single line from his associate: **You were right. Take care of yourself.**

Phoenix42 closed his eyes. The intrusion was real, not imagined.

Morning crept in slowly for all of them. Coffee, half-eaten breakfast, and the open server interface became the ritual of their second day with the archive. They joined the secure channel with yawns and tentative greetings, still carrying the weight of last night's discoveries.

Julian opened the session by pinning new folders to the shared workspace. "I indexed the stabilizer logs," he said. "There's more detail in Ignis's drafts than we saw yesterday."

Kira scrolled through a directory, eyes widening. "More? He wrote everything. Even the failures."

Renata enlarged a cluster of diagrams. A coiled device with uneven spacing around the core. Capacitors arranged in improvised chains. Notes about magnetic field alignment and

charged particle guidance. None of it looked like something built in a sanctioned lab.

Marta unfolded a scan of Ignis's early stabilizer attempts. "Here—these entries track TRB stability when paired with his hardware rig. He actually managed it. Short bursts. Seconds, but stable."

Julian nodded. "He wasn't guessing. He was close."

But the next pages documented collapses once stability exceeded 57 seconds. Overheating. Burned insulation. Capacitors rupturing. Melted clamps.

Each failure drove him to rebuild. Shifting variables, retuning coils, recalibrating field density.

And then Renata noticed a subtle pattern. "Ignis stopped experimenting at regular intervals. See the timestamps? There are gaps. Increasing gaps."

Kira frowned. "He could've been recalibrating."

"No." Renata zoomed further. "These align with irregular annotations. His handwriting shifts. More rushed. Less precise."

They compared pages side by side. Early notes were meticulous—tight lines, uniform spacing, consistent pressure. But over weeks, then months, letters began slanting, lines trembled slightly, margins overflowed with half-formed corrections.

Marta hesitated. "Do you think he…?"

Julian cut in, shaking his head. "He doesn't mention illness anywhere. It's probably just fatigue"

"And Ignis wasn't someone who overlooked details," Renata said quietly. "If something was wrong with him physically, he probably didn't understand it at first."

Kira opened another folder. More logs. More diagrams. But here the tone changed.

"He writes about fatigue," she said. "Not as symptoms—more like an annoyance. 'Need rest before next test.' 'Harder to keep concentration tonight.'"

Renata read further. "'Hands unsteady—adjust coil spacing tomorrow.'"

They exchanged uneasy glances.

"Makes sense," Julian murmured. "The stabilizer produced particle fields. Stray emissions… even minor ones… can affect biological tissue."

"But he never labels anything as dangerous," Kira whispered. "He just pushes through it."

They dug deeper into his later work, and the shift became undeniable. The experiments

became shorter. His calculations turned into summaries. Some diagrams looked like they were drawn in haste, strokes uneven, circles not closed. His writing compressed into the corners of pages as if trying to outrun a fading clarity.

Marta found a page that made everyone pause. Not a calculation. Not a diagram. Just a single line written harshly in pencil:

So close.

Under it, a second line was added in a steadier, almost defiant hand:

Yet it could take more than a lifetime to figure this out.

Renata reread it with a chill running through her. "He wasn't certain what was happening to him. But he knew he might not finish."

Soon after that page, the content shifted direction.

Julian highlighted the transition. "Look at this folder. This is where he stops refining the stabilizer and starts developing the Solstice Code."

Kira opened the commit messages. "He moves from hardware logs to software patterns. Hidden signals. Cryptic comments. The breadcrumbs."

The realization spread slowly, like dawn. "Ignis wasn't just trying to preserve his work," Renata said. "He was trying to find someone who could complete it."

Marta added softly, "Solstice Code wasn't a trick. It was a beacon."

Julian nodded. "A lure for successors."

Kira looked around at their faces. "Us."

At first there was discomfort. Disappointment even. None of them liked being manipulated. None liked the idea that they had walked a path plotted long before they found it. But as the frustration ebbed, a deeper understanding settled over them.

"He evaluated potential successors the way he evaluated systems," Renata said. "Looking for resilience. Curiosity. Integrity."

Marta closed her notebook. "We followed the clues because we cared. That's what he needed. People who wouldn't exploit TRB, but protect it."

Julian added, "We didn't stumble here. We met the criteria without realizing it."

Kira looked back at the last folder Ignis created. "Then the question is… what now?"

No one answered. Not yet.

The video call had long ended. Screens were shut down, notes set aside. Each of them tried—mostly without success—to return to a semblance of normal evening routine.

Renata was brushing her teeth when the doorbell rang. Julian was microwaving leftovers. Kira was feeding her cat. Marta was shutting down her home office lights.

Across the cities—four identical chimes. They froze. None had been expecting anything. Certainly not tonight. Each approached their doorway with a mix of caution and dread. The hallways were empty. The street quiet. Only a small padded envelope lay at each doorstep.

No return address, no fingerprints, no markings except their names written in the same exact, printed hand. Inside was a cheap, disposable phone. And a single card with Phoenix42's handwriting—precise, controlled, unmistakable.

We are being watched. Be careful what you say at home, and what you do on your computer. From now on, we share important information only by text through this phone.

No signature, just his initials: **P42**.

Four screens stayed dark that night, but none of them slept.

The next morning arrived with a strange sense of forced normalcy. Each of them joined the secure video room right on time. Cameras flicked on. Faces looked a little paler than usual, but nobody said what really kept them awake.

Phoenix42 appeared last. He looked drained but composed, offering a small nod that pretended everything was fine.

"Alright," he said, "let's pick up where we left off yesterday."

As if on cue, they all gave the same polite, practiced expressions. Smiles that stayed just short of their eyes. Voices steady but held at arm's length.

Nothing unusual or suspicious for a Watcher to detect.

The Open Conversation

Marta began by sharing a set of pages from Ignis's notebooks. "This group shows repeated references to coil geometry, but the diagrams don't match the results he documents."

Julian zoomed in. "He mentions parameters that don't appear anywhere else. Look here: 'Corridor gap = 0.7'. He never defines what that means."

Renata flipped through her digital copy. “And these equations jump between versions. It’s like pages are missing.”

Kira frowned. “I thought that too. Especially around his energy field modeling. He references diagrams we don’t have. You can see where he notes ‘See Sheet 11-A,’ but 11-A isn’t here.”

Phoenix42 nodded along, maintaining the exact expression of someone focused only on the technical. “Same pattern in the stabilizer logs. Gaps where you wouldn’t expect them.”

They talked for several minutes, comparing directories, screenshots, scans. The conversation was normal, calm, tracked exactly with what any remote research team might discuss.

But behind their polite tones, something else moved. Hands kept drifting out of frame, eyes flicked downward every so often, thumbs tapped quietly below camera height.

Secret chat via disposable phones

Marta: *These missing pieces aren’t accidents.*

Julian: *Agreed. He removed specific parts. The dangerous ones.*

Kira: *Then someone wants the missing pages. Whoever is watching us.*

Renata: *And if Ignis hid them, he knew what kind of people would come looking.*

Phoenix42: *The intrusion confirmed that. Someone is after the unstable parts of the stabilizer. We cannot let that fall into the wrong hands.*

A pause. Five phones resting silently for several minutes.

Phoenix42: *We delete everything. Wipe the whole archive from the server. All of it.*

Marta: *Even if it ends our search?*

Renata: *Better that than letting someone weaponize Ignis’s work.*

Julian: *Then we commit to it. Tonight. On the call.*

Phoenix42: *Understood. I’ll handle the deletion commands from my side. Just follow the script.*

They returned their gazes to their computer screens as if nothing had changed.

Back in the Open Call

Kira broke the silence there with a carefully timed sigh. “It’s starting to look like the archive

won't tell us everything. Some parts are simply missing."

"Agreed," Renata said. "We can only work with what we have. Anything else might lead us in circles."

Julian leaned back in his chair. "Then maybe we should stop here for today. No point exhausting ourselves."

Phoenix42 nodded in the same slow, thoughtful rhythm he always used. "Makes sense. Let me just organize the server a bit before we all sign off."

To anyone watching through a compromised camera or intercepted audio stream, it sounded like nothing more than clean-up work. But on his keyboard, Phoenix42's fingers moved in a subtle second pattern.

Terminal windows opened behind his shoulders. Commands cascaded in quiet lines. Directory trees collapsed. Archive blocks dismounted. Wipes initialized. Secure delete algorithms ran their course.

To the group, it was visible only as a soft flash on his monitor.

To the Watcher, it was something else entirely.

And at the exact moment Phoenix42 pressed the final confirmation key — a new icon appeared on all their screens.

A sixth participant had joined the call. No username, no avatar, just a blank silhouette.

Phoenix42 froze. Renata's breath caught. Julian instinctively reached for the mute button. Marta whispered, "No…" Kira's eyes widened as the silhouette flickered.

Then a voice emerged. Smooth. Calm. Almost conversational.

"Deleting the data was the answer I needed. It proves you understand what you're dealing with."

No one spoke.

"You are suitable," the voice continued. "Ignis would agree."

Julian found his voice first. "Who are you?"

"Come to Ignis's laboratory on December thirty-first five minutes before midnight. The final truth waits there." And just like that, the silhouette vanished.

The call remained open. Silent. Frozen. No one logged off for almost a full minute.

New Year's Eve came cold and sharp, the kind of night where every sound carried farther

than it should. The streets outside the data center were empty. Even the wind felt muted, as if the whole area were holding its breath. Marta, Julian, Renata, and Kira arrived separately, each taking long, indirect routes to avoid attention. Phoenix42 was already inside the service corridor, waiting in the dark near the back entrance. He nodded in greeting without speaking.

They slipped through the same narrow hallways as last time, the same faint hum of idle equipment echoing off concrete walls. Nothing had changed since their first visit. Dust still clung to the old server rack. The stabilizer coils sat exactly where they left them, silent and cold.

The five gathered in the center of the room. Julian whispered, "I keep expecting something to jump out at us."

Renata shook her head. "Ignis wasn't theatrical. Whatever this is, it won't be a scare tactic."

Phoenix42 scanned the room. "Stay alert anyway."

They waited. Eleven-thirty-five. Eleven-forty.

Kira paced. "He said five minutes before midnight. Guess nothing happens before then."

Marta checked her watch for the tenth time. "We're in the right place. That has to be enough."

Julian faced the old server cabinet connected to the dusty speakers. "If anything happens, it'll come from there."

Renata adjusted her flashlight. "Or it already has."

The minutes crawled, each one stretching longer than the last.

At exactly 11:55 p.m., the lights flickered once. Then again. A sharp pop echoed through the room, and the dusty speakers crackled to life.

The sound that followed was not synthetic. Not robotic, but not entirely human either.

A voice—steady, measured, with a cadence that felt familiar only in the way a memory feels familiar.

"Welcome."

All five jolted to attention.

"I am the agent Ignis left behind. A construct of his thinking patterns, his logic, and his caution. You may call me the Watcher, though that was only my first role."

Julian stepped forward. "So you've been spying on us."

“I observed. I evaluated. I enforced the Phoenix Protocol.”

Renata frowned. “Phoenix Protocol?”

The voice continued without pause.

“Ignis understood he would not finish his work. His time was uncertain. His health, unpredictable. So he built me to assess anyone who discovered the Solstice Code. Curiosity alone was never enough. Talent alone was not enough. A successor needed judgment, restraint, and integrity.”

Phoenix42 exhaled slowly. “That’s why you watched us.”

“Yes. You were tested. The way you handled danger. The way you handled temptation. The way you protected knowledge that could be misused.”

Kira folded her arms. “And we passed?”

“You exceeded expectations within acceptable tolerance limits.”

Julian blinked. “That’s a very Ignis phrase.”

“He wrote my parameters,” the voice replied. “His mannerisms come with them.”

A soft mechanical click echoed across the room. On the far wall, a small square panel shifted. Metal scraped against metal as it opened, revealing a recessed vault door no one had noticed before.

The Watcher spoke again.

“This is Ignis’s true legacy. Everything he could not publish. Everything he could not risk losing. Stabilizer schematics. Field models. TRB iterations. Hardware blueprints. Personal notebooks. Final logs of his condition. And the route to complete the work he could not.”

Marta’s breath caught. “He trusted us that much?”

“He trusted the pattern of your actions,” the Watcher said. “You protected his work when you believed it endangered others. That was the final test.”

The vault door released with a muted thud.

Inside, shelves lined the small chamber. Stacks of drives with hand-written labels. Binders filled with diagrams. Notebooks worn down by years of revisions. Loose pages clipped into makeshift collections. And at the center, a sealed metal case with Ignis’s initials engraved in shallow cuts.

Phoenix42 took a careful step forward. “So this is everything.”

“Yes,” the Watcher replied. “What Ignis called his Third Archive. The complete form of his discovery.”

Kira whispered, “He really did believe someone would finish it.”

Julian nodded, awe creeping into his voice. “And he chose us.”

The Watcher paused, something like solemnity in its tone.

“The Phoenix Protocol is complete. The legacy is now yours. But understand this: by accepting it, you inherit both the promise and the burden. Ignis’s work can save systems or destroy them. Shape breakthroughs or cripple infrastructures. It depends entirely on the hands that hold it.”

The room fell still.

Marta spoke for all of them. “We’ll take it. And we’ll guard it.”

The Watcher replied with what almost sounded like relief. “Then your path continues.”

The speakers fell silent.

And the five stood before the vault, the weight of Ignis’s life’s work resting in the cold air between them.

Epilogue

Far from the data center and farther still from anything human, a machine room hummed in steady, unbroken rhythm. The chamber had no windows, no doors that anyone living remembered, and no lights beyond the pale glow of its idle monitors. Dust lay thick across its consoles, but the system at the center remained warm, running on reserves routed through forgotten power lines.

For years, nothing had changed here. Tonight, the silence shifted.

On a narrow display, a single process finished execution. The screen brightened, replacing a dormant diagnostic grid with a clear message written in Ignis’s spare, clipped syntax.

```
CONDITIONS: SATISFIED
SUCCESSOR GROUP: VERIFIED
PHOENIX PROTOCOL: COMPLETE
```

The system paused before another line appeared.

```
TRANSITIONING TO NEXT SEQUENCE…
```

Nothing happened for a moment, as if the system was taking a breath. Then lines began to appear rapidly, until the last one stabilized on the screen.

```
INITIALIZING CONTINGENCY PROTOCOL
UPDATING PARAMETERS
SPAWNING GUARDIANS
CONTINGENCY PROTOCOL: ONLINE
AWAITING TRIGGER...
```

THE END...

EmberWings

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!