https://doi.org/10.31891/2219-9365-2025-83-2
УДК 658:004.652:004.6

ЛИСАК Віктор
Хмельницький національний університет
https://orcid.org/0000-0001-5352-7090
email: lysak.viktor@khmnu.edu.ua
МИХАЛЬЧУК Ігор
Хмельницький національний університет
https://orcid.org/0009-0002-5162-5986
email: mykhalchukiv@khmnu.edu.ua

FIREBIRD ЯК СУБД ДЛЯ ІНФОРМАЦІЙНОЇ СИСТЕМИ ЗАКЛАДУ ВИЩОЇ ОСВІТИ: ПЕРЕВАГИ ТА ВИКЛИКИ, ПРАКТИЧНИЙ ДОСВІД

У статті представлено аналіз досвіду використання системи управління базами даних Firebird як основи інформаційної системи Хмельницького національного університету. Розглянуто етапи еволюції Firebird, архітектурні особливості та ключові нововведення у різних версіях СУБД. Особливу увагу приділено питанням інтеграції з сучасними вебсервісами, аналітичними платформами та мобільними додатками, а також проблемам сумісності з популярними фреймворками та ORM. Окреслено основні ризики безпеки та надано практичні рекомендації щодо забезпечення надійної роботи Firebird у закладах вищої освіти. Наведено поради щодо адміністрування, оновлення та захисту даних у великих академічних середовищах.

Ключові слова: Firebird, СУБД, інформаційна система, заклад вищої освіти, інтеграція, безпека, адміністрування, відкритий код, ORM, управління даними

LYSAK Viktor, MYKHALCHUK Ihor
Khmelnytskyi National University

FIREBIRD AS A DBMS FOR HIGHER EDUCATION INFORMATION SYSTEMS: BENEFITS, CHALLENGES AND IMPLEMENTATION EXPERIENCE

The research explores the architectural models of Firebird, including SuperServer, Classic, and Embedded, and highlights their significance for different types of applications within a university setting. Particular attention is devoted to comparing key features and innovations introduced in Firebird versions 2.5, 3.0, 4.0, and 5.0, such as improved transaction management, enhanced security mechanisms (including SRP authentication and encryption), native replication, and expanded support for modern data types.

The article investigates the main challenges of integrating Firebird with contemporary web services, analytical platforms, and mobile applications. Limitations in the availability of standardized REST/SOAP connectors, as well as the relatively basic support for JSON and NoSQL functionality compared to other open-source DBMSs (such as PostgreSQL or MySQL), are identified as significant barriers to rapid development and system interoperability. The compatibility of Firebird with popular programming frameworks and object-relational mapping (ORM) tools – such as Django, .NET, Java, Node.js, and PHP – is analyzed in detail. The authors observe that, although various drivers and adapters are available, integration may require additional effort and technical expertise.

Given the sensitivity of academic data, special emphasis is placed on information security. The paper reviews common vulnerabilities associated with SQL injection, brute-force attacks on authentication mechanisms, open network ports, outdated drivers, and misconfigured access rights. A set of practical recommendations is provided for mitigating these risks, based on the authors' extensive experience in the administration and modernization of the university's information systems.

In conclusion, the article provides a set of guidelines for developers and system administrators, emphasizing the importance of continuous monitoring, regular software updates, effective backup management, and adherence to best security practices. The findings highlight Firebird's advantages and limitations as a DBMS for higher education institutions. Additionally, the article offers valuable insights for those considering its implementation or upgrade in large-scale academic environments.

Keywords: Firebird, DBMS, information system, higher education, integration, security, administration, open source, ORM, data management

Стаття надійшла до редакції / Received 22.06.2025
Прийнята до друку / Accepted 19.07.2025

ПОСТАНОВКА ПРОБЛЕМИ У ЗАГАЛЬНОМУ ВИГЛЯДІ ТА ЇЇ ЗВ'ЯЗОК ІЗ ВАЖЛИВИМИ НАУКОВИМИ ЧИ ПРАКТИЧНИМИ ЗАВДАННЯМИ

Сучасні заклади вищої освіти (ЗВО) дедалі активніше впроваджують інформаційні системи для автоматизації управлінських і навчальних процесів з метою підвищення якості надання освітніх послуг, підвищення управління освітнім та науковим процесом. Важливу роль у таких системах відіграють системи управління базами даних (СУБД), від вибору та правильного впровадження яких залежать масштабованість, надійність, безпека і довгострокові перспективи розвитку інформаційної інфраструктури.

Firebird як відкрита реляційна СУБД вже понад два десятиліття використовується в різних сферах, зокрема і в освітньому середовищі та демонструє стійкість, продуктивність і доступність для розробників різного рівня.

Інформаційна система «Електронний університет» Хмельницького національного університету є прикладом успішного багаторічного впровадження Firebird, що дозволяє проаналізувати реальні переваги та недоліки цієї СУБД в умовах функціонування сучасного ЗВО.

Разом із зростанням вимог до інтеграції з зовнішніми сервісами, мобільними платформами та аналітичними інструментами, а також підвищенням вимог до інформаційної безпеки, гостро постає питання оптимізації, оновлення та підтримки таких систем на основі Firebird.

Актуальність теми дослідження зумовлена потребою у комплексному аналізі можливостей, викликів та практичних аспектів застосування Firebird у IT-інфраструктурах ЗВО.

ФОРМУВАННЯ ЦІЛЕЙ СТАТТІ

Метою статті є визначення основних переваг і викликів використання СУБД Firebird для інформаційної системи ЗВО, аналіз інтеграційних та безпекових аспектів її впровадження, а також формулювання практичних рекомендацій щодо експлуатації, адміністрування та оновлення Firebird.

ОСНОВНИЙ МАТЕРІАЛ ДОСЛІДЖЕНЬ

Інформаційна система Хмельницького національного університету «Електронний університет» вже багато років є ефективним засобом цифровізації управління навчальним процесом. Однією із основних її складових є Firebird, яку було обрано в якості центральної СУБД системи.

Динамічний розвиток апаратно-програмного інструментарію постійно вимагає від розробників цієї системи аналізувати потреби користувачів і підлаштовуватися до різноманітних викликів сьогодення. Як наслідок, – необхідно постійно вдосконалювати структуру СУБД, оптимізувати запити і процедури, оновлювати системне програмне забезпечення щоб отримувати вигоду від використання оновлених версій програмних продуктів.

IT-інфраструктура Хмельницького національного університету складається з різноманітних СУБД, оскільки до неї інтегровані різні інформаційні комплекси, бази знань, інструменти і засоби.

Еволюція та архітектурні особливості СУБД Firebird. У 2000-х роках, коли розпочиналося проєктування інформаційної системи «Електронний університет», було прийнято рішення використати СУБД Firebird. В той час СУБД Firebird виникла внаслідок відкриття вихідного коду InterBase, розробленої компанією Borland у 1980-х роках. У 2000 році, після опублікування коду InterBase 6.0 й була створена спільнота розробників Firebird, яка розпочала незалежний розвиток системи [1, 5]. Протягом більш ніж двадцятирічної історії Firebird пройшла декілька важливих етапів еволюції (рис. 1).

Рис. 1. Еволюція СУБД Firebird (2000–2025 рр.)

Примітка: сформовано авторами на основі [1, 3, 5]

Етап стабілізації (Firebird 1.0–1.5, 2002–2006 рр.) характеризувався усуненням спадкових обмежень InterBase, впровадженням багатоплатформної підтримки та стабілізацією базових механізмів транзакційності.

Етап модернізації (Firebird 2.0–2.5, 2006–2011 рр.) ознаменувався переходом до сучасніших стандартів SQL, розширенням функціоналу через нові типи даних і тригери, появою різних архітектурних моделей (SuperServer, Classic, Embedded) та значним покращенням підтримки багатопроцесорних систем [1, 5].

Етап архітектурної реконструкції (Firebird 3.0, 2011–2021 р.) передбачав докорінну перебудову внутрішньої архітектури з переходом до модульного ядра, впровадження сучасних механізмів аутентифікації (SRP), розширення підтримки стандарту SQL:2011 та появу пакетів [1, 3, 5].

Етап функціональної диверсифікації (Firebird 4.0–5.0…, з 2021 і по теперішній час) спрямований на підвищення гнучкості та масштабованості через нові типи даних (DECFLOAT, INT128, UUID), підтримку часових поясів, вдосконалену реплікацію та плагінів ядра (engine plugins) [1, 22].

Важливою особливістю Firebird є підтримка кількох архітектурних моделей роботи з підключеннями та процесами, що дозволяє адаптувати СУБД під специфіку конкретного проєкту (рис. 2).

Модель SuperServer передбачає використання єдиного серверного процесу для обробки всіх підключень, що забезпечує оптимальну роботу для середніх і великих систем завдяки централізованому управлінню кешем та ефективному використанню ресурсів.

Функціонування моделі Classic базується на створенні окремого процесу для кожного підключення,. Такий підхід забезпечує кращу масштабованість на багатопроцесорних серверах та підвищену відмовостійкість.

Модель вбудованого застосування (Embedded) призначена для додатків та десктопних рішень, де СУБД функціонує як бібліотека в рамках основного процесу додатка.

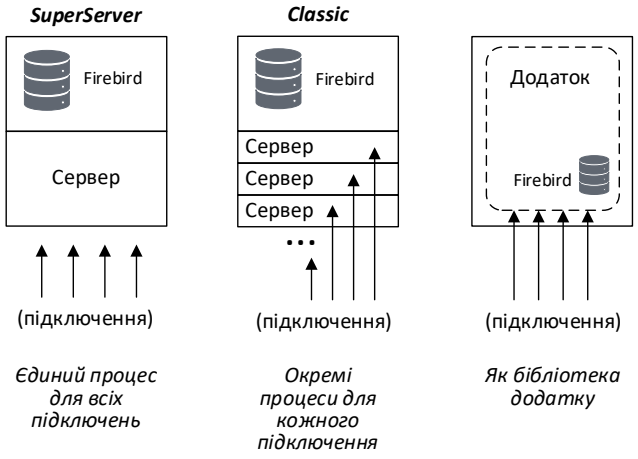

Рис. 2. Архітектурні моделі Firebird

Примітка: сформовано авторами на основі [1, 5]

З моменту появи Firebird розробники отримали можливість працювати з відкритою, масштабованою та надійною СУБД, яка поступово розширювала свій функціонал, підтримку стандартів SQL та інструменти для адміністрування. Кожна нова версія містила низку важливих змін, які суттєво впливали як на можливості розробників, так і на вимоги до експлуатації та обслуговування системи.

Для комплексного розуміння еволюції СУБД Firebird ми систематизували найважливіші відмінності між основними версіями у вигляді таблиці 2.

Таблиця 2.

Порівняння основних відмінностей між флагманськими версіями Firebird

Версія	Рік випуску	Основні нововведення	Підтримка ОС	Сумісність
2.5	2010	Покращена багатопоточність, підтримка симетричної багатопроцесорної обробки, нові SQL-функції	Windows, Linux, Mac OS	Висока
3.0	2016	Модульна архітектура, SRP-аутентифікація, пакети, SQL:2011		Часткова
4.0	2021	DECFLOAT/INT128/UUID, часові пояси, нативна реплікація		Обмежена
5.0	2023	Плагіни ядра, нові SQL-оператори, розширена реплікація		

Примітка: сформовано авторами на основі [1, 3, 5, 22]

Таким чином, здійснений нами аналіз ключових нововведень у флагманських версіях Firebird дозволив узагальнити їхні основні особливості на кожному важливому етапі розвитку можливостей цієї СУБД.

Firebird 2.5 закріпила позиції СУБД у корпоративному сегменті завдяки стабільній роботі та ефективній підтримці симетричної багатопроцесорної обробки. Основними її досягненнями стали удосконалені механізми управління транзакціями та розширення набору SQL-функцій.

Firebird 3.0 стала переломним моментом у розвитку системи у зв'язку з переходом до модульної архітектури, що спростило оновлення та розширення функціоналу. Впровадження протоколу SRP (Secure Remote Password) значно підвищило безпеку аутентифікації, а підтримка пакетів та розширень SQL:2011 (включно з віконними функціями та CTE-виразами, покращенням у тригерах і процедурах) розширила можливості розробників.

Firebird 4.0 зосередилася на підтримці сучасних типів даних та підвищенні гнучкості. Додавання типів DECFLOAT, INT128, UUID та повноцінної підтримки часових поясів відповідала потребам глобальних розподілених систем. Одним із найважливіших нововведень стало впровадження нативної реплікації, що дозволило створювати відмовостійкі рішення, оскільки резервне копіювання могло відбуватися вже у реальному часі і полегшило масштабування систем [22].

Firebird 5.0 продовжила розвиток у напрямку гнучкості через додаткові плагіни ядра, нові SQL-оператори та вдосконалення механізмів реплікації та моніторингу, що є особливо актуальним для високонавантажених корпоративних систем [5].

Окрім того, з кожною новою версією зростала роль дотримання стандартів безпеки, актуальності клієнтських бібліотек та необхідності детального тестування під час міграції.

Для резервного копіювання та відновлення Firebird традиційно використовує утиліти gbak (логічний рівень) та nbackup (фізичний рівень), які розвивалися з кожною версією та отримали додаткові функції для забезпечення цілісності даних.

Починаючи з версії 3.0, у Firebird суттєво розширилися можливості моніторингу. Зокрема, з'явилися відповідні розширені таблиці, які дозволили відстежувати активні підключення, транзакції, запити, індекси тощо. Окрім того, розробники удосконалили механізм протоколювання і додали гнучкі налаштування журналів помилок та діагностичних повідомлень.

Також, з кожною новою версією покращувалася підтримка сторонніх інструментів для адміністрування та розробки.

Практичний досвід використання СУБД Firebird у Хмельницькому національному університеті. СУБД Firebird забезпечує функціонування основної частини управлінської та освітньої діяльності в інформаційній системі «Електронний університет». Вона інтегрує в собі дані відділу кадрів, канцелярії, деканатів, навчального відділу, електронних кабінетів студентів та викладачів.

За багато років функціонування адміністратори та розробники університет тестували та використовували у роботі різні інструменти для роботи з СУБД Firebird. Деякі з них до теперішнього часу використовуються у нашій роботі. Кожний з таких засобів займає певну нішу і використовується для автоматизації певних видів робіт, а на це є різні причини, розуміння яких можливе з аналізу доступного інструментарію екосистеми Firebird.

Нижче, нами здійснено аналіз основних інструментів для роботи з базами даних Firebird та представлено результати у вигляді таблиці 3.

Таблиця 3.

Основні засоби адміністрування та розробки для СУБД Firebird від сторонніх розробників

Інструмент	Тип	Підтримувані операційні системи	Основні функції	Ліцензія	Особливості
FlameRobin	Вільне кросплатформне середовище розробки	Linux, Windows, macOS	- Візуальне середовище для розробки та адміністрування баз даних Firebird	Безкоштовна (відкритий код)	- Підтримує кілька платформ - Простий інтерфейс для базових завдань
IBExpert	GUI-оболонка для розробки та адміністрування	Windows	- Підтримка кількох версій Firebird та InterBase - Редактори для всіх об'єктів БД - Налагоджувач процедур і тригерів - Експорт даних	Платна (з безкоштовною версією для країн колишнього СРСР)	- Потужний набір інструментів для розробників - Підтримка старих версій Firebird
SQL Management Studio for InterBase/Firebird	Комплекс додатків для розробки та адміністрування	Windows	- Адміністрування баз даних - Міграція і пошук баз даних - Побудова SQL-запитів - Імпорт, експорт і порівняння даних	Платна	- Інтегроване середовище для повного циклу роботи з базами даних
DBeaver	Універсальний інструмент для баз даних	Windows, Linux, macOS	- Підтримка багатьох баз даних, включаючи Firebird - Візуальне проектування БД - Управління даними	Безкоштовна (відкритий код)	- Підтримує кілька баз даних - Зручний для розробників, які працюють з різними СУБД
Database Tour	Інструмент для роботи з базами даних	Windows	- Відкриття баз даних через різні інтерфейси (FD, Interbase, ADO) - Підтримка SQL-запитів	Платна	- Гнучкість у способах підключення до бази даних - Підходить для досвідчених користувачів
Firebird Development Studio	Студія розробки	Windows	- Розробка та адміністрування баз даних Firebird	Платна	- Спеціалізована для Firebird - Включає інструменти для розробки та управління

З таблиці 3 видно, що кожний інструмент має унікальні переваги: FlameRobin підходить для початківців, IBExpert і SQL Management Studio для професіоналів, DBeaver – для роботи з кількома СУБД, а Database Tour і Firebird Development Studio – для досвідчених користувачів.

Проблеми інтеграції та сумісності. Сучасні інформаційні системи ЗВО часто інтегруються з зовнішніми API, BI-платформами, хмарними та мобільними рішеннями. Впровадження Firebird як основної СУБД для таких систем супроводжується низкою викликів: обмежена підтримка стандартних протоколів інтеграції, відсутність повноцінної підтримки JSON/NoSQL-функціоналу, потреба у драйверах і проміжному програмному забезпеченні, обмежене використання у розробці мобільних застосунків, підтримка сучасних ORM і популярних фреймворків [1, 5, 10, 12].

Обмежена підтримка стандартних протоколів інтеграції. На відміну від таких популярних СУБД, як PostgreSQL чи MySQL, екосистема готових REST/SOAP-з'єднувачів для Firebird є значно вужчою, що і є причиною певних викликів для розробників, які прагнуть швидко інтегрувати цю базу даних із сучасними веб-додатками. Наприклад, PostgreSQL має інструменти типу PostgREST, які дозволяють автоматично формувати REST API на основі структури бази даних, а MySQL підтримується великою кількістю популярних бібліотек і фреймворків для легкого створення API. Для Firebird таких універсальних інструментів бракує і розробникам доводиться покладатися на власні рішення або адаптувати вже наявні бібліотеки. Хоча Firebird підтримує стандартні драйвери, такі як Jaybird для Java, node-firebird для JavaScript або fdb для Python, формування REST чи SOAP API вимагає додаткових зусиль з інтеграції цих драйверів із відповідними фреймворками [7, 8, 9]. Серед прикладів можна навести такі як Spring Boot, FastAPI чи Express.js для REST, або JAX-WS і WCF для SOAP.

Той факт, що екосистема Firebird є не дуже насиченою, по-перше, зумовлена меншою популярністю даної СУБД у порівнянні з PostgreSQL чи MySQL. А по-друге, він пояснюється також орієнтацією спільноти Firebird на стабільність і продуктивність самої СУБД, а не на розробку готових інструментів для API. Зокрема, комерційний проєкт HQbird є розширеним дистрибутивом Firebird і пропонує певні можливості для створення REST API через плагіни, такі як MySQLEngine чи ODBCEngine, але ці засоби потребують додаткового налаштування і не є настільки універсальними, як аналоги для інших баз даних [11].

Такий підхід, хоча і є гнучким, проте ускладнює швидке розгортання стандартних API, особливо у проєктах, коли бюджет обмежений, а терміни реалізації є порівняно невеликими. Для порівняння, у PostgreSQL чи MySQL розробник може скористатися готовими рішеннями, які значно прискорюють інтеграцію з вебдодатками чи мобільними платформами. У випадку з СУБД Firebird, попри її надійність і підтримку сучасних функцій, відсутність широкої екосистеми готових з'єднувачів змушує розробників інвестувати додаткові ресурси у створення власних рішень. Це може бути виправданим для проєктів, де Firebird уже є основною СУБД, але створює бар'єри для нових користувачів, які шукають простіші способи інтеграції через REST чи SOAP [7].

Відсутність повноцінної підтримки JSON/NoSQL-функціоналу. Наявність потужного інструментарію для роботи з JSON-даними є важливою. В першу чергу, сучасні front-end фреймворки, такі як React чи Angular, орієнтовані на роботу зі структурованими JSON-даними, які є продуктом належної роботи API-засобів. По-друге, аналітичні платформи типу Power BI чи Tableau, ефективніше працюють із базами даних, що підтримують гнучкі запити до JSON.

Відсутність повноцінної підтримки JSON/NoSQL-функціоналу у Firebird, незважаючи навіть на появу підтримки типу даних JSON у версії 4.0 і вище, залишається суттєвим недоліком цієї СУБД у порівнянні з конкурентами. І такі обмеження як раз і формують виклики при інтеграції СУБД Firebird з сучасними front-end платформами. У Firebird 4.0 з'явився тип даних JSON, що дозволяє зберігати дані у форматі JSON як текстові рядки, а також були додані базові функції для їх обробки, такі як JSON_QUERY і JSON_VALUE. Однак ці можливості є досить обмеженими: Firebird не підтримує складні операції з JSON, наприклад, індексацію полів усередині JSON-структур чи виконання агрегаційних запитів. Це означає, що розробникам необхідно або обробляти JSON-дані на стороні клієнтського додатка, або створювати додаткові SQL-запити для розбору текстових даних, що й ускладнює розробку [1, 5].

У той же час інші СУБД, наприклад PostgreSQL, пропонує потужний функціонал для роботи з JSONB: індексацію GIN/GiST, підтримку повнотекстового пошуку та можливість використання JSON у складних аналітичних запитах.

У Firebird розробникам доводиться або обмежуватися базовими операціями з JSON, або використовувати зовнішні інструменти для перетворення даних у потрібний формат, що додає складності та може вимагати додаткових ресурсів на сервері чи клієнтській стороні.

Також обмежена підтримка JSON/NoSQL-функціоналу ускладнює використання Firebird у проєктах, де потрібна гнучкість NoSQL-підходів, наприклад, у системах із динамічними схемами даних або у вебдодатках, які вимагають швидкої адаптації до змін у структурі даних. Хоча Firebird залишається надійною реляційною СУБД із сильними сторонами, такими як компактність і висока продуктивність у транзакційних системах, її відставання у підтримці сучасних форматів даних, таких як JSON, робить її менш конкурентоспроможною у порівнянні з PostgreSQL чи навіть MongoDB у контексті інтеграції з новими

технологіями [3]. Для проєктів, де важлива гнучка робота з JSON, розробникам часто доводиться або компенсувати ці недоліки додатковим кодом, або розглядати альтернативні СУБД.

Потреба з драйверами та проміжним програмним забезпеченням. Інтеграція СУБД Firebird з іншими сервісами також є нелегким завданням для розробників, оскільки часто необхідно використовувати спеціалізовані "шлюзи" або проміжні конвертери даних. На відміну від таких СУБД, як PostgreSQL чи MySQL, які широко підтримуються стандартними драйверами і бібліотеками, Firebird потребує ретельного підбору інструментів щоб забезпечити стабільне підключення до бази даних. Серед таких інструментів найбільш відомими є Jaybird для Java, node-firebird для JavaScript та fdb для Python [7, 8, 9]. Ці драйвери є поширеними, але у багатьох випадках можна спостерігати певні обмеження у їхній функціональності або такі засоби часто вимагають додаткових налаштувань для інтеграції з сучасними сервісами. Внаслідок цього розробники часто змушені створювати власні сервіси за допомогою проміжного програмного забезпечення, наприклад, через поєднання можливостей веб-серверів Apache/Nginx та скриптів на PHP/Python, або ж розробляти власні API з нуля коли практична реалізація стає трудомісткою і вимагає додаткових знань як СУБД Firebird, так і технологій веброзробки.

Крім того, сумісність версій драйверів із конкретними версіями СУБД Firebird є критично важливою. Наприклад, Jaybird 6.0.2 підтримує Firebird 5.0.3, але використання старіших версій драйверів із новішими випусками СУБД може призводити до появи помилок або зниження продуктивності. Через те розробникам необхідно постійно відстежувати актуальність клієнтських бібліотек і їх оновлень, що особливо є актуальним для проєктів з тривалим життєвим циклом. Відповідно відсутність автоматичних інструментів для спрощення таких інтеграцій, на відміну від екосистем PostgreSQL чи MySQL, змушує розробників Firebird витрачати додаткові ресурси на тестування та налатування.

У свою чергу, така залежність від спеціалізованих драйверів і проміжного ПЗ ускладнює інтеграцію СУБД Firebird із хмарними платформами, аналітичними системами та сучасними фронтенд-додатками, які передбачають використання стандартизованих протоколів і форматів даних, таких як JSON. Наприклад, для інтеграції з аналітичними платформами, такими як Power BI, може знадобитися створення власних з'єднувачів або використання ODBC-драйверів Firebird, які також потребують клопіткого налаштування та перевірки сумісності. У порівнянні з конкурентами, де такі інтеграції часто автоматизовані або підтримуються готовими інструментами, Firebird вимагає від розробників глибшого розуміння як самої СУБД, так і екосистеми проміжного ПЗ, що стає перепоною для швидкого розгортання проєктів.

Обмежене використання у мобільних застосунках. Обмеження Firebird щодо використання в мобільних застосунках значною мірою зумовлені відсутністю «легкого» режиму, подібного до SQLite, що робить цю СУБД менш придатною для роботи як локальна база даних на мобільних клієнтах. На відміну від SQLite, яка спеціально розроблена для вбудованих систем і мобільних пристроїв завдяки своїй компактності, мінімальним вимогам до ресурсів і здатності працювати без серверного процесу, Firebird потребує значно більше ресурсів і складнішого налаштування. Наприклад, Firebird зазвичай працює в серверному режимі (Classic, SuperServer або Embedded), і навіть Embedded-версія, хоча й призначена для локального використання, не є настільки легкою, як SQLite, через залежність від бібліотек і необхідність конфігурації. Це робить її непрактичною для прямого використання на пристроях з обмеженими ресурсами, таких як смартфони чи планшети, де SQLite залишається стандартом завдяки своїй простоті й ефективності.

Оптимальним сценарієм застосування Firebird у контексті мобільних застосунків є використання її як серверної СУБД з організацією доступу через API. Наприклад, Firebird може ефективно працювати на сервері для обробки транзакційних запитів, зберігання великих обсягів даних або виконання складних SQL-операцій, тоді як мобільний клієнт взаємодіє з базою через REST або SOAP API, створені за допомогою фреймворків, таких як Spring Boot, FastAPI чи Express.js. Такий підхід дозволяє використовувати сильні сторони Firebird, такі як надійність, підтримка транзакцій ACID і висока продуктивність у багатопотокових середовищах, але вимагає додаткової інфраструктури для створення та підтримки API. Наприклад, для інтеграції з мобільними додатками розробники можуть використовувати Jaybird (для Java) або node-firebird (для JavaScript) для створення серверних API, які передають дані у форматі JSON, зручному для фронтенду мобільних платформ, таких як React Native чи Flutter.

Ця залежність від серверної інфраструктури та API створює додаткові виклики, особливо для невеликих проєктів, де швидке розгортання локальної бази даних на клієнтському пристрої є пріоритетом. У порівнянні з SQLite, яка дозволяє зберігати дані безпосередньо на пристрої без необхідності серверного компонента, Firebird вимагає стабільного мережевого з'єднання для доступу до сервера, що може бути проблематичним у сценаріях із нестабільним інтернетом. Крім того, створення API для Firebird потребує ретельного управління драйверами та їх сумісністю, як зазначено в попередніх матеріалах, що додає складності розробці. Таким чином, хоча Firebird є потужною СУБД для серверних застосунків, її обмеження в мобільних контекстах змушують розробників або використовувати альтернативні бази даних для клієнтської сторони, або інвестувати в розробку надійних серверних API для забезпечення інтеграції.

Підтримка сучасних ORM і популярних фреймворків. Підтримка Firebird у сучасних ORM (Object-Relational Mapping) і популярних фреймворках залишається обмеженою порівняно з іншими

відкритими СУБД, такими як PostgreSQL чи MySQL, що ускладнює розробникам прагнення інтегрувати цю базу даних у сучасні веб- та мобільні додатки. Нижче детально розглянемо підтримку Firebird у різних мовах програмування та фреймворках, а також пов'язані з цим труднощі.

У Python-екосистемі підтримка СУБД Firebird обмежена, особливо в популярних фреймворках типу Django. Офіційно Django не підтримує Firebird як одну зі стандартних баз даних. Проте існує сторонній адаптер django-firebird, який дозволяє інтегрувати Firebird із Django, але його підтримка часто відстає від актуальних версій фреймворку. Це змушує розробників або самостійно доопрацьовувати адаптер, або обмежувати функціонал проєкту та відмовлятися від використання нових можливостей Django. Окрім того, документація та спільнота django-firebird є досить скромною, що й ускладнює вирішення проблем і пошук готових рішень [15].

Інший популярний інструмент для Python – SQL Alchemy, підтримує Firebird через драйвер FDB і забезпечує базову інтеграцію. Проте робота з Firebird у SQLAlchemy вимагає додаткових налаштувань, особливо для специфічних типів даних, таких як BLOB чи JSON. Як ми вже згадували, підтримка JSON у Firebird є базовою, і SQLAlchemy не завжди може ефективно обробляти такі дані без додаткових мапінгів. Окрім того, розробникам необхідно тестувати основні запити чи транзакції, оскільки їх виконання через FDB може бути неоптимізованим порівняно з драйверами для PostgreSQL чи MySQL. Фактично, це дозволяє використовувати SQLAlchemy, але не є ідеальним рішенням для роботи з СУБД Firebird, особливо в проєктах із високими вимогами до продуктивності [16].

У .NET-екосистемі Firebird має стабільний офіційний драйвер FirebirdClient, який підтримує ADO.NET і дозволяє інтегрувати базу даних із популярними ORM, такими як Entity Framework (EF) і Dapper. FirebirdClient забезпечує підключення до Firebird, але його оновлення часто відстають від релізів нових версій СУБД. Через те, підтримка нових можливостей СУБД може з'являтися в драйвері із затримкою. Entity Framework Core, хоча й підтримує Firebird через FirebirdClient, також потребує додаткових налаштувань для коректної роботи зі специфічними типами даних або складними транзакціями. Зокрема, реалізація складних відношень «багато-до-багатьох» чи асинхронних операцій може вимагати побудови власних мапінгів або різноманітних «хитрощів» для обходу обмежень [9].

Dapper, як мікро-ORM, є менш вимогливим до драйверів, але також залежить від «якості» реалізації FirebirdClient. Хоча Dapper дозволяє швидко виконувати SQL-запити, але всеодно розробникам доводиться вручну враховувати особливості Firebird, такі як робота з JSON чи специфіку транзакцій. У порівнянні з PostgreSQL, де Entity Framework Core має офіційну підтримку з розширеним функціоналом, Firebird залишається обділеною СУБД і менш гнучкою опцією для .NET-розробників, оскільки вимагає додаткових зусиль для інтеграції.

У Java-екосистемі Firebird підтримується через Jaybird, який є офіційним JDBC-драйвером, що дозволяє інтегрувати базу даних із популярними фреймворками такими як Hibernate і Spring Data JPA. Jaybird є відносно стабільним з підтримкою останніх версій Firebird. Проте підтримка нових можливостей СУБД (тип даних JSON, покращення індексації), ще не повністю доступна в Hibernate, оскільки мапінг нових типів даних потребує оновлення діалекту Hibernate для Firebird. Такі відставання у розробці драйверів для Firebird і ускладнює розробку порівняно з PostgreSQL, де JSONB підтримується «з коробки» [7].

Spring Data JPA, який часто використовується разом із Hibernate, також залежить від можливостей, підтримуваних драйвером Jaybird. Хоча Spring Boot дозволяє створювати REST API для Firebird розробникам всеодно доводиться враховувати обмеження драйвера, такі як підтримка специфічних функцій Firebird (тригери, збережені процедури). У порівнянні з MySQL чи PostgreSQL, де інтеграція із Spring є більш зрілою, СУБД Firebird потребує додаткових зусиль для забезпечення повної сумісності, особливо в проєктах із використанням складних ORM-функцій кешування чи асинхронних запитів.

У Node.js-екосистемі підтримка Firebird є особливо слабкою. Популярні ORM, такі як Sequelize і TypeORM, не мають офіційної підтримки Firebird, а такі сторонні бібліотеки як node-firebird, часто обмежені за функціоналом і не завжди вчасно оновлюються для роботи з новими версіями СУБД. Наприклад, node-firebird дозволяє виконувати SQL-запити, але не забезпечує повноцінного ORM-функціоналу, як Sequelize для PostgreSQL чи MySQL. Це означає, що розробники змушені писати багато «сирого» SQL-коду або створювати власні абстракції, що знижує продуктивність процесу розробки. TypeORM теоретично може бути адаптований для Firebird через кастомні драйвери, проте також потребує значних зусиль та робить його непрактичним вибором. У результаті Node.js-розробники часто уникають СУБД Firebird та віддають перевагу базам даних із кращою підтримкою ORM [8].

У PHP-екосистемі Firebird також не має офіційної підтримки в популярних фреймворках, таких як Laravel. Laravel підтримує PostgreSQL, MySQL і SQLite власними засобами, але для Firebird потрібні сторонні адаптери, такі як laravel-firebird. Ці адаптери часто мають певні обмеження, наприклад, несумісність із новими версіями Laravel або відсутність підтримки певних функцій, таких як міграції чи складні відношення в Eloquent ORM. Інший популярний PHP ORM – Doctrine, може працювати з Firebird через драйвер PDO Firebird, але його конфігурація є складнішою, ніж для інших СУБД, і потребує додаткових зусиль для обробки специфічних типів даних Firebird, таких як BLOB чи JSON [17].

Таким чином, підтримка Firebird у сучасних ORM і популярних фреймворках значно поступається іншим відкритим СУБД, таким як PostgreSQL чи MySQL, через обмежену кількість офіційних адаптерів, затримки в оновленні драйверів і менш активну спільноту. Розробникам доводиться витрачати додаткові ресурси на налаштування, тестування сумісності та створення власних рішень для інтеграції Firebird із сучасними технологіями. Хоча стабільні драйвери, такі як Jaybird і FirebirdClient, забезпечують базову функціональність, все одно їхні обмеження в підтримці нових можливостей СУБД і складність інтеграції з ORM роблять Firebird менш привабливим вибором для проєктів, де потрібна швидка розробка з використанням популярних фреймворків. Для максимальної ефективності розробникам необхідно ретельно тестувати інтеграцію, використовувати актуальні версії драйверів і, за можливості, комбінувати Firebird із легшими базами даних для клієнтської сторони.

Безпекові виклики при інтеграції СУБД в інформаційну систему ЗВО. Безпека інформаційних систем, зокрема тих, що використовуються у ЗВО є надзвичайно важливою, оскільки вони часто містять конфіденційні дані. Firebird, як реляційна СУБД, має як загальні вразливості, притаманні всім базам даних, так і специфічні ризики, пов'язані з її архітектурою та конфігурацією. Хоча Firebird пропонує надійні механізми безпеки, такі як підтримка SRP-аутентифікації з версії 3.0, її ефективність залежить від правильного налаштування, використання актуальних версій драйверів і дотримання найкращих практик [19]. Розглянемо основні вразливості Firebird, їх причини та рекомендації щодо їх уникнення, щоб забезпечити надійний захист даних у ЗВО.

Однією з найпоширеніших загроз для Firebird є SQL-ін'єкції, які виникають, коли запити до бази даних формуються динамічно без використання параметризації. Якщо додаток дозволяє включати введені користувачем дані безпосередньо в рядки SQL-запитів, то зловмисник може вставити шкідливий код, який змінить логіку запиту або надасть несанкціонований доступ до даних. Firebird, як і інші СУБД, не має вбудованого захисту від таких атак, якщо розробники не дотримуються безпечних практик програмування [20].

Інший тип безпекових ризиків – брутфорс-атака до SRP-аутентифікації. Починаючи з версії 3.0 у Firebird було запроваджено механізм Secure Remote Password (SRP), який значно підвищив безпеку аутентифікації у порівнянні з попередніми версіями, де паролі зберігалися у менш захищеному вигляді. SRP використовує криптографічні методи для захисту паролів, але якщо користувачі або адміністратори встановлюють слабкі паролі, то система залишається вразливою до брутфорс-атак. Зловмисники можуть намагатися підібрати паролі шляхом багаторазових спроб підключення, особливо якщо сервер СУБД доступний через відкритий порт [19].

Потрібно також пам'ятати, що СУБД Firebird за замовчуванням використовує порт 3050 для підключень і якщо сервер неправильно налаштований, то цей порт може бути відкритим для всіх IP-адрес, що створює ризики несанкціонованого доступу. Окрім того, неправильні налаштування файлу firebird.conf або мережевих налаштувань також можуть дозволити зловмисникам сканувати сервер та отримувати доступ до бази даних або запускати атаки DDoS-типу [18, 21].

Варто пам'ятати, що використання у проєктах застарілих версій драйверів, таких як Jaybird, FirebirdClient або node-firebird, може сформувати вразливості, оскільки старі бібліотеки можуть містити невиправлені помилки безпеки або не підтримувати нові функції, такі як SRP-аутентифікація чи шифрування.

Окрім того, неправильне налаштування прав доступу у Firebird може дозволити користувачам або ролям отримувати більше привілеїв, аніж необхідно, що створює ризики несанкціонованого доступу до даних або їх модифікації.

Загальні рекомендації. Зважаючи, на багаторічний досвід використання СУБД Firebird в інформаційній інфраструктурі Хмельницького національного університету, хочемо виділити певні рекомендації для розробників та адміністраторів СУБД Firebird.

Для запобігання SQL-ін'єкцій слід дотримуватися таких підходів до розробки:

– обов'язкове використання параметризованих запитів або підготовлених виразів (prepared statements) у драйверах, таких як Jaybird (Java), FirebirdClient (.NET) або node-firebird (Node.js);

– недопущення конкатенації SQL-рядків із даними, введеними користувачем;

– використання ORM (типу SQLAlchemy або Entity Framework), які автоматично застосовують параметризацію;

– регулярне проведення аналізу коду та використання інструментів статичного аналізу для виявлення потенційних вразливостей у SQL-запитах.

Для захисту від брутфорс-атак слід виділити наступні заходи:

– використання складних паролів довжиною не менше 12 символів, які складаються з літер, цифр та спеціальних символів;

– встановлення обмежень на кількість спроб підключення на рівні мережі, через брандмауер або подібні інструменти, типу fail2ban, для блокування IP-адрес після певної кількості невдалих спроб авторизації;

– регулярний аудит логів підключень для виявлення підозрілої активності;

– використання додаткових механізмів аутентифікації, таких як інтеграція з LDAP або Active Directory через плагіни Firebird.

Для контролю доступу до СУБД необхідно:

– дозволити підключення лише з довірених IP-адрес із використанням брандмауера або параметру RemoteBindAddress у файлі firebird.conf.

– використовувати VPN або SSH-тунелювання для доступу до сервера Firebird із віддалених клієнтів з метою забезпечення додаткового рівня шифрування;

– здійснювати регулярний аудит конфігурації мережі та списків доступу згідно політики безпеки;

– увімкнути шифрування даних на рівні мережі (Firebird 3.0+ і плагіни шифрування).

Для забезпечення стабільного підключення та належної роботи бази даних потрібно:

– регулярне оновлювати клієнтські бібліотеки до останніх версій, які рекомендуються спільнотою Firebird Project;

– перевіряти сумісність драйверів із використовуваною версією Firebird перед оновленням СУБД, щоб уникнути проблем із підключенням;

– відслідковувати оновлення на офіційному сайті Firebird та в репозиторіях відповідних драйверів;

– застосувати автоматизовані інструменти для моніторингу оновлень бібліотек.

Щоб знизити безпекові ризи, нами рекомендується:

– використання принципу найменших привілеїв: надавати ролям і користувачам лише ті права, які необхідні для виконання їхніх завдань;

– проведення регулярного аудит користувачів і їхніх привілеїв за допомогою SQL-запитів до системних таблиць типу RDB$USER_PRIVILEGES.

– обмеження доступу до адміністративної ролі SYSDBA і використання складних паролів для цього облікового запису;

– використання інструменту gsec для управління користувачами та їхніми правами.

Окрім того, варто виділити і специфічні для СУБД Firebird поради розробникам і адміністраторам.

По-перше, для забезпечення надійної роботи СУБД і підтримання оптимальної продуктивності необхідно регулярно використовувати моніторингові інструменти Firebird, аналізувати плани виконання запитів та проводити технічне обслуговування бази даних. Особливу увагу слід приділяти моніторингу "старих" транзакцій, стану “очищувача сміття” та ефективності використання індексів. Використовуйте утиліти gbak та nbackup для створення повних та інкрементальних копій. Регулярно перевіряйте працездатність резервних копій шляхом відновлення на окремому сервері.

По-друге, успішна експлуатація та міграція інформаційних систем на основі Firebird вимагають дотримання низки технічних та організаційних рекомендацій, які враховують як особливості окремих версій, так і загальні принципи забезпечення надійності та продуктивності роботи СУБД. При роботі з Firebird рекомендуємо уникати міграцій з пропуском версії та оновлюватися послідовно, що дозволить виявляти потенційні проблеми на ранніх етапах та забезпечити стабільність роботи системи.

По-третє, перед оновленням варто ознайомитися з розділами «release notes», переліком змін (changelog) та можливими особливостями переходу з поточної версії. Це дозволить своєчасно виявити проблеми із несумісністю та потенційно критичні зміни у системі. Оцініть критичні відмінності між версіями. Особливу увагу приділяйте змінам у типах даних, механізмах аутентифікації, поведінці збережених процедур, роботі індексів та особливостям реплікації. Оновлюйте драйвери, ORM і бібліотеки, що працюють з Firebird, після переходу на нову версію. Це особливо важливо при впровадженні нових типів даних, зміні протоколів аутентифікації чи підтримці реплікації. Перед оновленням продуктивної бази даних доцільно розгорнути тестову копію, виконати міграцію на ній і перевірити працездатність всіх критичних модулів та звітів.

По-четверте, необхідно забезпечити, щоб у коді застосунків не залишалися відкриті транзакції, навіть лише для читання. Це стосується як вебдодатків, так і клієнтських програм.

По-п’яте, необхідно мінімізувати використання застарілих UDF і віддавати перевагу вбудованим, оскільки з появою нових версій СУБД більшість поширених функцій реалізовані у самому ядрі Firebird. Регулярно переглядайте плани виконання складних SQL-запитів, видаляйте непотрібні або надлишкові індекси.

На нашу думку, дотримання вищевказаних рекомендацій сприяє підвищенню надійності роботи інформаційних систем, мінімізації ризиків під час оновлення версій Firebird і забезпеченню максимальної продуктивності навіть у складних сценаріях використання.

ВИСНОВКИ З ДАНОГО ДОСЛІДЖЕННЯ І ПЕРСПЕКТИВИ ПОДАЛЬШИХ РОЗВІДОК У ДАНОМУ НАПРЯМІ

Проведений аналіз досвіду впровадження та експлуатації СУБД Firebird в інформаційній системі Хмельницького національного університету дозволив виокремити її основні переваги у сучасному цифровому середовищі та виклики з якими стикаються розробники для цієї бази даних. Firebird продемонструвала високу надійність, стабільність роботи, відповідність вимогам до транзакційної цілісності та доступність для

адміністрування навіть у складних розподілених IT-інфраструктурах. Завдяки відкритому коду та активній спільноті, система постійно розвивається, що дозволяє адаптувати її під актуальні вимоги ЗВО.

Разом з тим, результати дослідження свідчать, що інтеграція Firebird із сучасними вебсервісами, мобільними та аналітичними платформами ускладнюється через обмежену підтримку стандартних протоколів обміну даними, базову реалізацію JSON/NoSQL-функціоналу та недостатню кількість універсальних драйверів і з'єднувачів. Виявлено, що робота з ORM і популярними фреймворками часто вимагає додаткових зусиль з боку розробників та адміністраторів, що може впливати на швидкість впровадження нових функцій у системі.

На теперішній час особливу увагу слід приділяти питанням інформаційної безпеки, зокрема захисту від SQL-ін'єкцій, брутфорс-атак, належному управлінню правами доступу і оновленню драйверів та клієнтських бібліотек. Дотримання рекомендованих практик адміністрування, моніторингу та резервного копіювання дозволяє мінімізувати ризики і підвищити ефективність роботи інформаційної системи на базі СУБД Firebird.

Таким чином, СУБД Firebird може розглядатися як ефективне рішення для побудови інформаційних систем ЗВО, особливо для завдань з високими вимогами до надійності та транзакційності. Однак для реалізації комплексних інтеграційних проєктів і швидкої адаптації до сучасних технологій доцільно враховувати обмеження цієї платформи та завчасно планувати додаткові ресурси на супровід і розробку.

Література

1. Borrie H. The Firebird Book: A Reference for Database Developers / H. Borrie. – 2nd ed. – Berkeley: Apress, 2004. – 1104 p.
2. Beach P. Migration to Firebird 3: Step by Step / P. Beach // Firebird Conference Proceedings. – Prague, 2017. – P. 45–67.
3. Rodriguez M. Transaction Management in Firebird: Best Practices / M. Rodriguez // Database Systems Journal. – 2022. – Vol. 13, No. 2. – P. 34–51.
4. Firebird SQL Reference. Firebird 4.0 Language Reference / Firebird Documentation Team. – 2021. – [Електронний ресурс]. – Режим доступу: https://firebirdsql.org/file/documentation/chunk/en/refdocs/fblangref40/fblangref40.html – Дата звернення: 03.04.2025.
5. FirebirdSQL Project. Official Documentation. – [Електронний ресурс]. – Режим доступу: https://firebirdsql.org/en/documentation/ – Дата звернення: 17.05.2025.
6. Firebird FAQ. – [Електронний ресурс]. – Режим доступу: https://www.firebirdfaq.org/ – Дата звернення: 15.03.2025.
7. Jaybird Documentation. – [Електронний ресурс]. – Режим доступу: https://firebirdsql.org/en/jdbc-driver/ – Дата звернення: 21.04.2025.
8. Node-firebird: GitHub repository. – [Електронний ресурс]. – Режим доступу: https://github.com/hgourvest/node-firebird – Дата звернення: 09.03.2025.
9. Firebird ODBC Driver. – [Електронний ресурс]. – Режим доступу: https://firebirdsql.org/en/odbc-driver/ – Дата звернення: 19.03.2025.
10. PostgreSQL JSON Documentation. – [Електронний ресурс]. – Режим доступу: https://www.postgresql.org/docs/current/datatype-json.html – Дата звернення: 12.03.2025.
11. HQbird Documentation. – [Електронний ресурс]. – Режим доступу: https://ibase.ru/hqbird/ – Дата звернення: 05.05.2025.
12. Firebird JSON Support. – [Електронний ресурс]. – Режим доступу: https://firebirdsql.org/en/devel-json/ – Дата звернення: 18.04.2025.
13. Firebird Embedded Documentation. – [Електронний ресурс]. – Режим доступу: https://firebirdsql.org/en/firebird-embedded/ – Дата звернення: 25.03.2025.
14. SQLite Documentation. – [Електронний ресурс]. – Режим доступу: https://www.sqlite.org/docs.html – Дата звернення: 01.05.2025.
15. Django-firebird: GitHub repository. – [Електронний ресурс]. – Режим доступу: https://github.com/maxirobaina/django-firebird – Дата звернення: 05.03.2025.
16. SQLAlchemy FDB Driver. – [Електронний ресурс]. – Режим доступу: https://pypi.org/project/fdb/ – Дата звернення: 13.04.2025.
17. Laravel-firebird: GitHub repository. – [Електронний ресурс]. – Режим доступу: https://github.com/jacquestvanzuydam/laravel-firebird – Дата звернення: 07.04.2025.
18. Firebird Security Guide. – [Електронний ресурс]. – Режим доступу: https://firebirdsql.org/file/documentation/html/en/firebirddocs/sec/firebird-security.html – Дата звернення: 18.05.2025.

19. Firebird SRP Authentication. – [Електронний ресурс]. – Режим доступу: https://firebirdsql.org/file/documentation/html/en/refdocs/fblangref40/firebird-40-security.html – Дата звернення: 06.03.2025.
20. SQL Injection Prevention. – [Електронний ресурс]. – Режим доступу: https://www.owasp.org/index.php/SQL_Injection_Prevention_Cheat_Sheet – Дата звернення: 10.03.2025.
21. Firebird Configuration Guide. – [Електронний ресурс]. – Режим доступу: https://firebirdsql.org/file/documentation/reference_manuals/user_manuals/html/qsg30-config.html – Дата звернення: 20.03.2025.
22. Firebird 4.0 Release Notes. – [Електронний ресурс]. – Режим доступу: https://firebirdsql.org/file/documentation/release_notes/html/en/4_0/rnfb40.html – Дата звернення: 11.04.2025.

References

1. Borrie, H. (2004). *The Firebird Book: A Reference for Database Developers* (2nd ed.). Berkeley: Apress. 1104 p.
2. Beach, P. (2017). Migration to Firebird 3: Step by Step. In: *Firebird Conference Proceedings*, Prague, pp. 45–67.
3. Rodriguez, M. (2022). Transaction Management in Firebird: Best Practices. *Database Systems Journal*, 13(2), 34–51.
4. Firebird Documentation Team. (2021). *Firebird 4.0 Language Reference*. Available at: https://firebirdsql.org/file/documentation/chunk/en/refdocs/fblangref40/fblangref40.html (Accessed: 03 April 2025).
5. FirebirdSQL Project. Official Documentation. Available at: https://firebirdsql.org/en/documentation/ (Accessed: 17 May 2025).
6. Firebird FAQ. Available at: https://www.firebirdfaq.org/ (Accessed: 15 March 2025).
7. Jaybird Documentation. Available at: https://firebirdsql.org/en/jdbc-driver/ (Accessed: 21 April 2025).
8. Node-firebird: GitHub repository. Available at: https://github.com/hgourvest/node-firebird (Accessed: 09 March 2025).
9. Firebird ODBC Driver. Available at: https://firebirdsql.org/en/odbc-driver/ (Accessed: 19 March 2025).
10. PostgreSQL JSON Documentation. Available at: https://www.postgresql.org/docs/current/datatype-json.html (Accessed: 12 March 2025).
11. HQbird Documentation. Available at: https://ibase.ru/hqbird/ (Accessed: 05 May 2025).
12. Firebird JSON Support. Available at: https://firebirdsql.org/en/devel-json/ (Accessed: 18 April 2025).
13. Firebird Embedded Documentation. Available at: https://firebirdsql.org/en/firebird-embedded/ (Accessed: 25 March 2025).
14. SQLite Documentation. Available at: https://www.sqlite.org/docs.html (Accessed: 01 May 2025).
15. Django-firebird: GitHub repository. Available at: https://github.com/maxirobaina/django-firebird (Accessed: 05 March 2025).
16. SQLAlchemy FDB Driver. Available at: https://pypi.org/project/fdb/ (Accessed: 13 April 2025).
17. Laravel-firebird: GitHub repository. Available at: https://github.com/jacquestvanzuydam/laravel-firebird (Accessed: 07 April 2025).
18. Firebird Security Guide. Available at: https://firebirdsql.org/file/documentation/html/en/firebirddocs/sec/firebird-security.html (Accessed: 18 May 2025).
19. Firebird SRP Authentication. Available at: https://firebirdsql.org/file/documentation/html/en/refdocs/fblangref40/firebird-40-security.html (Accessed: 06 March 2025).
20. SQL Injection Prevention. Available at: https://www.owasp.org/index.php/SQL_Injection_Prevention_Cheat_Sheet (Accessed: 10 March 2025).
21. Firebird Configuration Guide. Available at: https://firebirdsql.org/file/documentation/reference_manuals/user_manuals/html/qsg30-config.html (Accessed: 20 March 2025).
22. Firebird 4.0 Release Notes. Available at: https://firebirdsql.org/file/documentation/release_notes/html/en/4_0/rnfb40.html (Accessed: 11 April 2025).