JOURNAL
COMPUTER SCIENCES INSTITUTE

JCSI 31 (2024) 82–88
Received: 21 February 2024
Accepted: 18 March 2024

# Comparative analysis of database types in mobile applications running on the Android operating system

# Analiza porównawcza typów baz danych w aplikacjach mobilnych działających na systemie operacyjnym Android

Adrian Bartoszek*

*Department of Computer Science, Lublin University of Technology, Nadbystrzycka 36B, 20-618 Lublin, Poland*

**Abstract**
The article focused on conducting a detailed comparative analysis of the different types of databases used in mobile applications on the Android platform. The research included four main types of databases: relational (SQLite), key-value (SnappyDB), object-oriented (ObjectBox) and non-relational (Couchbase Lite). The analysis focused on four key operations: writing, reading, editing and deleting data. In addition, the work evaluated each database type in terms of resource consumption, such as RAM and CPU. This research aimed to provide a comprehensive perspective on the performance and efficiency of each database type in the context of mobile applications on the Android platform. The research confirmed that the ObjectBox database has better performance than the other analyzed databases.

*Keywords*: sqlite; snappydb; objectbox; couchbase lite

**Streszczenie**
Artykuł skupił się na przeprowadzeniu szczegółowej analizy porównawczej różnych typów baz danych wykorzystywanych w aplikacjach na platformie Android. Badania obejmowały cztery główne rodzaje baz danych: relacyjna (SQLite), klucz-wartość (SnappyDB), obiektowa (ObjectBox) i nierelacyjna (Couchbase Lite). Analiza skupiła się na czterech operacjach: zapisie, odczycie, edycji i usuwaniu danych. Dodatkowo, artykuł dokonał oceny poszczególnych typów baz danych pod kątem zużycia pamięci RAM oraz procesora. Badania te miały na celu dostarczenie kompleksowej perspektywy dotyczącej wydajności i efektywności każdego typu bazy danych w kontekście aplikacji działających na platformie Android. Badania potwierdziły, że baza danych ObjectBox ma lepszą wydajność niż pozostałe analizowane bazy danych.

*Słowa kluczowe*: sqlite; snappydb; objectbox; couchbase lite

*Corresponding author
*Email address*: adrian.bartoszek@pollub.edu.pl (A. Bartoszek)



## 1. Wstęp

Dynamiczny rozwój rynku aplikacji mobilnych stanowi niezaprzeczalny element współczesnej cyfrowej rewolucji. Android jest aktualnie najpopularniejszym systemem operacyjnym posiadającym udział wynoszący 41.61% w rynku systemów operacyjnych na świecie [1]. Rynek aplikacji mobilnych nieustannie się rozwija i rośnie. Obecnie Android posiada liczbę aktywnych użytkowników wynoszącą ponad 2,5 miliarda w ponad 190 krajach [2]. Współczesne aplikacje mobilne często muszą działać na różnorodnych urządzeniach, zarówno tych bardziej zaawansowanych, jak i tych o ograniczonych zasobach. Optymalny dobór bazy danych wpływa znacząco na korzystanie z aplikacji przez użytkowników końcowych, którzy oczekują płynnego działania aplikacji nawet na starszych urządzeniach. Odpowiednio dobrany typ bazy danych przekłada się na szybkość wykonywania podstawowych operacji zapisu, odczytu, aktualizacji i usuwania danych, ale również na efektywność pod kątem zużycia zasobów np. pamięci RAM i procesora.

Niniejszy artykuł skupia się na bezserwerowych, wbudowanych bazach danych (embedded databases), które są częścią danej aplikacji, co oznacza, że działają wewnątrz samej aplikacji, a nie na dedykowanych serwerach. Wbudowane bazy danych nie muszą polegać na zewnętrznych zależnościach i zapewniają płynne działanie oraz minimalną konfigurację [3].

Wbudowane bazy danych dzięki eliminacji warstwy komunikacji klient-serwer, mogą zwiększyć swoją wydajność i zmniejszyć złożoność. Ich łatwość implementacji i niewielkie wymagania dotyczące obsługi sprawiają, że są wygodne w użytkowaniu i nie wymagają dużego wsparcia. Ponadto, posiadają niewielki rozmiar, co czyni je idealnymi dla urządzeń mobilnych z ograniczoną pamięcią. Dobra wbudowana baza danych powinna być elastyczna, aby móc obsłużyć duże ilości danych i dostosować się do zmieniających się potrzeb nowych urządzeń, a także powinna dobrze działać na starszych urządzeniach ze słabszymi, przestarzałymi podzespołami [4].

Głównym zagadnieniem niniejszego artykułu jest edukacyjne podejście do analizy istotnych aspektów związanych z podstawowymi operacjami baz danych, tj. zapisywaniem, odczytywaniem, edycją i usuwaniem danych, w kontekście aplikacji mobilnych na platformie Android. Celem tego badania jest zgłębienie, w jaki sposób różne typy baz danych radzą sobie z powyższymi operacjami pod względem efektywności i wydajności aplikacji oceniając także zużycie kluczowych podzespołów takich jak pamięć RAM i procesor.

## 2. Przegląd literatury

W artykule [5] przeprowadzono analizę porównawczą mobilnych baz danych dostępnych dla systemu Android. Porównano bazę danych SQLite, która jest relacyjną bazą, z nierelacyjnymi bazami Realm, ObjectBox i SnappyDB. Badania obejmowały różne aspekty, takie jak: zapis, edycja, usuwanie, wyszukiwanie i sortowanie danych. Wyniki badań pokazały, że w przypadku operacji CRUD (Create, Read, Update, Delete), wszystkie badane bazy wykazywały podobne czasy działania przy małej liczbie rekordów (poniżej 1000). Jednak przy operacjach na dużej liczbie rekordów, baza Realm osiągnęła najgorsze wyniki. Natomiast relacyjna baza SQLite była najszybsza w operacjach edycji rekordów. Baza Realm wykazała się znacznie lepszą wydajnością w operacjach wyszukiwania danych w porównaniu do bazy SQLite. Jeśli chodzi o złożone zapytania, baza SQLite i ObjectBox osiągnęły najlepsze wyniki, choć SQLite było nieco szybsze. Operacja sortowania była najszybsza w bazie Realm, a różnice między nią a innymi technologiami były znaczne.

W artykule [6] skupiono się na analizie wydajności systemu zarządzania relacyjną bazą danych SQLite na platformie Android. Artykuł przeprowadza analizę wydajności SQLite w kilku scenariuszach, obejmując operacje CRUD na danych niezaszyfrowanych, danych zaszyfrowanych oraz dostęp współbieżny do bazy danych. W tym celu opracowano i udostępniono aplikację o nazwie DBInspector na platformie Google Play Store, która demonstruje wpływ na wydajność bazy danych SQLite. Aplikacja DBInspector umożliwia przeprowadzanie operacji CRUD na danych niezaszyfrowanych oraz zaszyfrowanych, prezentując czasy wykonania każdej operacji. Aplikacja została udostępniona na platformie Google Play Store, dzięki czemu każdy może z niej skorzystać. Wyniki przeprowadzonych testów wskazują na różnice w wydajności SQLite w zależności od rodzaju danych i operacji. Operacje na danych zaszyfrowanych zajmują znacznie więcej czasu niż na danych niezaszyfrowanych. Ponadto, wyniki pokazują, że operacje INSERT i SELECT zajmują znacznie więcej czasu niż UPDATE i DELETE w przypadku danych niezaszyfrowanych. Jednakże, w przypadku danych zaszyfrowanych, wyniki są prawie odwrotne. Dodatkowo, testy dotyczące współbieżności wykazują, że nowy mechanizm blokowania SQLite zapewnia współbieżność z efektywnością wynoszącą od 40% do 44%.

Celem artykułu [7] było wykonanie analizy porównawczej baz danych: SQLite, ObjectBox, Realm, Room, greeenDAO. Room i greenDAO opierają się na SQLite natomiast ObjectBox i Realm przyjmują obiektowe, nierelacyjne podejście, które nie wymaga wiedzy z zakresu SQL. Wyniki analizy wykazały, że Room i greenDAO mają najniższą wydajność oraz nierelacyjne bazy danych takie jak ObjectBox i Realm zapewniają najlepszą wydajność pod względem czasu odpowiedzi w zakresie podstawowych operacji CRUD. Wyniki wyraźnie sugerują, że proste operacje są znacznie lepsze przy korzystaniu z baz danych nierelacyjnych. Autorzy podkreślają, że indeksowanie przyspiesza zapytania. Jednakże, zwiększenie liczby indeksów w tabeli spowoduje, że operacje takie jak zapis i aktualizacja zajmą więcej czasu, ponieważ te indeksy zostaną automatycznie zaktualizowane, co spowoduje spowolnienie wydajności w tych operacjach.

Artykuł [8] przeprowadza ocenę wydajności i łatwości implementacji następujących baz danych: SQLite, SnappyDB i Realm. Testy wykazały, że trzy powyższe rozwiązania mają zbliżoną łatwość implementacji oraz zużycie pamięci. Jednak w zakresie operacji zapisu i odczytu dużej liczby rekordów rozwiązanie SQLite wygenerowało dużo słabsze wyniki od Realm i SnappyDB. Wyniki także pokazują, że SQLite wykazuje najgorszą wydajność, podczas gdy Realm zapewnia najbardziej intuicyjny sposób dopasowania danych do logiki aplikacji ze względu na swoją obiektową naturę. Niemniej jednak według autorów SQLite pozostaje lepszym wyborem na potrzeby aplikacji z mniejszymi zbiorami danych.

## 3. Badane technologie

SQLite jest autonomiczną biblioteką działającą bezserwerowo. Biblioteka ta korzysta z transakcyjnego silnika bazodanowego SQL. Kod źródłowy jest dostępny dla wszystkich i jest darmowy do użytku dla każdego zarówno w celach prywatnych jak i komercyjnych. Projekt SQLite został rozpoczęty w dniu 09.05.2000 r. i intencją twórców jest wspieranie SQLite do roku 2050 [9].

SnappyDB jest otwarto źródłową bazą danych typu klucz-wartość dedykowaną dla systemu Android dla deweloperów, którzy chcą zastosować podejście nierelacyjne, alternatywne dla SQLite. Baza danych umożliwia przechowywanie prymitywnych typów danych, a także obiektów lub tablic w bezpieczny dla typów sposób. SnappyDB opiera się na bazie danych LevelDB i używa algorytmu kompresji snappy [10].

ObjectBox to baza danych stworzona między innymi z myślą o aplikacjach napisanych w językach Java i Kotlin. Pozwala łatwo przechowywać i zarządzać danymi w aplikacjach na platformie Android lub w środowisku JVM (Java Virtual Machine) na systemach Linux, macOS czy Windows. ObjectBox wyróżnia się intuicyjnością obsługi, umożliwiając efektywne składowanie danych w aplikacjach mobilnych oraz na różnorodnych platformach. Dzięki temu, programiści mają możliwość sprawnego korzystania z tej bazy danych [11].

Couchbase Lite to nierelacyjna, wbudowana baza danych w przechowująca dane w postaci dokumentów w formacie JSON, przeznaczona dla aplikacji mobilnych. Ta nierelacyjna baza danych umożliwia przechowywanie danych lokalnie i oferuje funkcje takie jak: możliwość tworzenia zapytań z semantyką opartą na SQL, pełno tekstowe wyszukiwanie dokumentów przechowywanych lokalnie oraz przechowywanie załączników do dokumentów, na przykład obrazów czy plików PDF. Couchbase Lite może również opcjonalnie pracować w relacji klient-serwer zapewniając synchronizację do chmury [12].

## 4. Stanowisko badawcze

Aplikacja testowa została stworzona korzystając z języka programowania Kotlin przy użyciu środowiska

programistycznego Android Studio Hedgehog w wersji 2023.1.1 Patch2. Aplikacja korzysta z biblioteki Android Faker, która pozwala na generowanie pseudolosowych danych imitujących dane rzeczywiste. Pozwala to na uruchamianie testów z danymi przypominającymi dane rzeczywiste, które są przechowywane w bazach danych. Każdy rekord zawierał wygenerowane dane wykorzystując powyższą bibliotekę takie jak: imię, nazwisko, adres e-mail, numer telefonu, adres, opis.

Interfejs aplikacji testowej obejmuje zestaw elementów umożliwiających użytkownikowi określenie parametrów testu. Zaliczają się do nich:

1. Pole wyboru typu radio button do określenia rodzaju bazy danych (SQLite, SnappyDB, ObjectBox, Couchbase Lite).
2. Lista rozwijana do określenia rodzaju testu (Create, Read, Update, Delete).
3. Pole typu EditText umożliwiające wprowadzenie liczby testów.
4. Pole typu EditText umożliwiające wprowadzenie liczby rekordów dla każdego testu.
5. Dwa przyciski: przycisk rozpoczęcia testu oraz przycisk czyszczenia pola z wynikami.

Zapewnienie takiego interfejsu pozwala na elastyczne dostosowanie parametrów testu oraz ułatwia interakcję użytkownika z aplikacją testową. Rysunek 1 przedstawia aplikację testową z przykładowymi wynikami.

Wybór konkretnych wersji bazy danych do testów zapewnia, że aplikacja korzystając z danej wersji bazy danych pracuje optymalnie i bez błędów. Wersje testowanych baz danych prezentują się następująco:

- ObjectBox 3.7.1,
- Couchbase Lite 3.1.1,
- SnappyDB 0.5.2,
- SQLite 3.28.0.

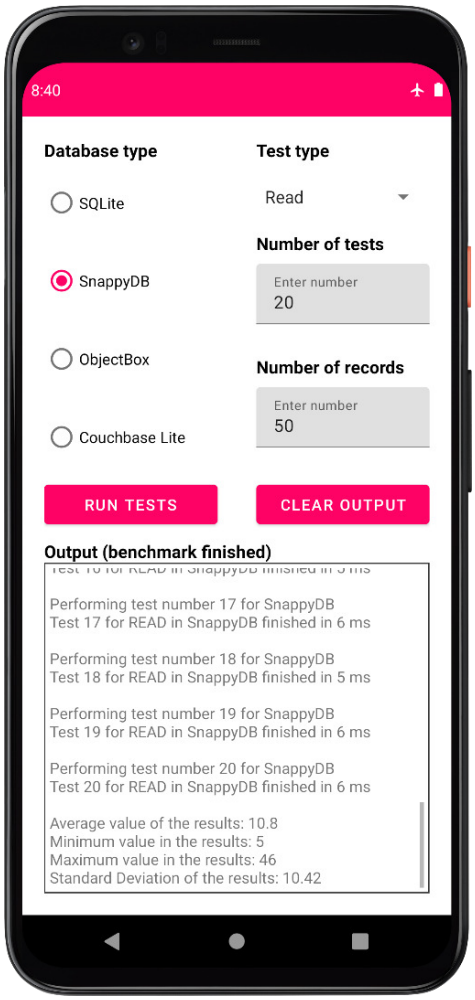
W celu uzyskania rzetelnych wyników, aplikacja została zainstalowana na dwóch różnych urządzeniach. Tabela 1 przedstawia parametry techniczne pierwszego urządzenia użytego do testów – Samsung Galaxy S10, natomiast tabela 2 przedstawia parametry drugiego urządzenia – Samsung Galaxy S21.

Tabela 1: Parametry techniczne Samsung Galaxy S10

| Parametr | Wartość |
|---|---|
| Procesor | Samsung Exynos 9820<br>8-rdzeniowy<br>z taktowaniem:<br>8 x 2.70 GHz |
| Pamięć RAM | 8 GB |
| Pamięć wbudowana | 128 GB |
| Wersja systemu | Android 13 |

Tabela 2: Parametry techniczne Samsung Galaxy S21

| Parametr | Wartość |
|---|---|
| Procesor | Samsung Exynos 2100<br>8-rdzeniowy<br>z taktowaniem:<br>1 x 2.9GHz,<br>3 x 2.8GHz,<br>4 x 2.2GHz |
| Pamięć RAM | 8 GB |
| Pamięć wbudowana | 128 GB |
| Wersja systemu | Android 14 |




Rysunek 1: Aplikacja testowa.

## 5. Metody badawcze

Badania przeprowadzone zostały dla czterech podstawowych operacji: zapisu, odczytu, aktualizacji i usuwania danych. Dla każdej z tych operacji przeprowadzono testy z użyciem różnej liczby rekordów: 100, 1000, 10000 oraz 100000. Aby zagwarantować rzetelność wyników, wszystkie testy zostały przeprowadzone na dwóch urządzeniach z systemem Android posiadających podzespoły o różnych parametrach (Samsung Galaxy S10 oraz Samsung Galaxy S21).

W trakcie trwania testów, urządzenia były utrzymywane w trybie samolotowym, co miało na celu zminimalizowanie ewentualnego przesyłania danych w tle, co mogłoby wpłynąć na wyniki testów. Dodatkowo, wszystkie aplikacje działające w tle zostały wyłączone, aby zapewnić spójność warunków testowych oraz minimalizować wpływ innych procesów na wydajność systemu. Ekran urządzenia był włączony przez cały czas testów, a urządzenie było podłączone do zasilania, aby zapewnić

stabilność działania i uniknąć ewentualnego wpływu wyładowania baterii na wyniki testów.

Na potrzeby przeprowadzenia badań opracowane zostały poniższe scenariusze badawcze:

1. Scenariusz badawczy S1 - przeprowadzenie operacji zapisu, odczytu, aktualizacji oraz usuwania danych dla 100 rekordów – badanie czasu wykonania.
2. Scenariusz badawczy S2 - przeprowadzenie operacji zapisu, odczytu, aktualizacji oraz usuwania danych dla 1000 rekordów – badanie czasu wykonania.
3. Scenariusz badawczy S3 - przeprowadzenie operacji zapisu, odczytu, aktualizacji oraz usuwania danych dla 10000 rekordów – badanie czasu wykonania.
4. Scenariusz badawczy S4 - przeprowadzenie operacji zapisu, odczytu, aktualizacji oraz usuwania danych dla 100000 rekordów – badanie czasu wykonania.
5. Scenariusz badawczy S5 - przeprowadzenie operacji zapisu dla 100000 rekordów – badanie zużycia procesora.
6. Scenariusz badawczy S6 - przeprowadzenie operacji zapisu dla 100000 rekordów – badanie zużycia pamięci RAM.

Test zapisu danych polegał na utworzeniu określonej liczby rekordów w bazie danych. Test odczytu danych polegał na odczytaniu określonej liczby rekordów z bazy danych i zwrócenie ich w formie listy. Test aktualizacji danych polegał na zaktualizowaniu określonej liczby rekordów w bazie danych. Test usuwania danych polegał na usunięciu określonej liczby rekordów z bazy danych.

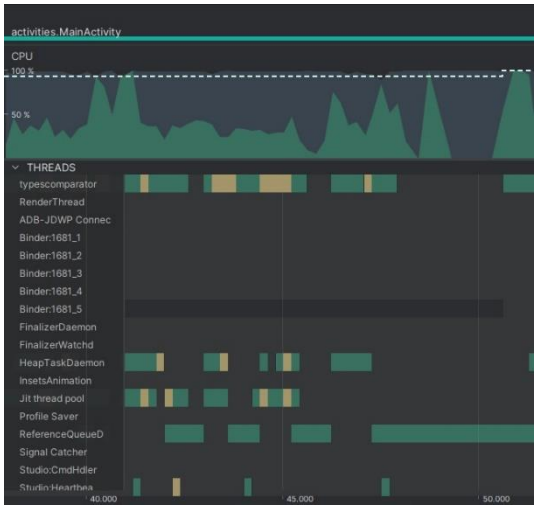
Wyniki czasu wykonania operacji w scenariuszach S1-S4 były uzyskiwane poprzez mierzenie czasu za pomocą metody currentTimeMillis() bezpośrednio przed i po wykonaniu danej operacji, a następnie obliczanie różnicy między tymi dwoma pomiarami. Warto zaznaczyć, że pomiar ten dotyczył jedynie czasu wykonania poszczególnych operacji, a nie procesów dotyczących generowania danych pseudolosowych czy innych czynności mogących wpłynąć na wyniki. Dodatkowo, w scenariuszach S5 i S6 zużycie pamięci RAM oraz procesora było monitorowane za pomocą odpowiednich narzędzi dostępnych w środowisku Android Studio, takich jak Memory Profiler i CPU Profiler.

Każdy scenariusz badawczy został powtórzony 20 razy w celu uzyskania wiarygodnych i reprezentatywnych wyników. Powtórzenie testów pozwoliło na eliminację ewentualnych anomalii czy przypadkowych zakłóceń oraz umożliwiło uzyskanie średniej wartości, co pozwoliło na lepszą ocenę wydajności systemu w różnych warunkach. Ponadto, powtarzanie testów pozwoliło na sprawdzenie stabilności wyników. W kontekście wszystkich przeprowadzonych scenariuszy badawczych, uwzględnione zostały średnie wartości, które bazowały na wynikach uzyskanych z 20 powtórzonych testów dla każdego scenariusza.

Android CPU Profiler jest narzędziem deweloperskim umożliwiającym analizę zużycia procesora przez aplikację na platformie Android. Pozwala programistom monitorować wykorzystanie zasobów procesora, śledzić wywołania metod oraz identyfikować potencjalne źródła opóźnień w aplikacji. Android Memory Profiler to narzędzie umożliwiające analizę zużycia pamięci przez aplikacje na platformie Android.

Rysunek 2 przedstawia przykładowe dane dotyczące zużycia pamięci RAM korzystając z narzędzia Memory Profiler, podczas gdy rysunek 3 przedstawia obciążenie procesora korzystając z narzędzia CPU Profiler. Dzięki tym narzędziom możliwe było dokładne zbadanie wpływu operacji zapisu danych na zasoby systemu.


Rysunek 2: Narzędzie Memory Profiler.

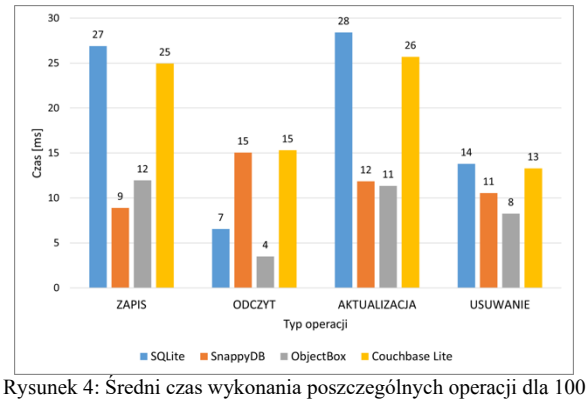



Rysunek 3: Narzędzie CPU Profiler.

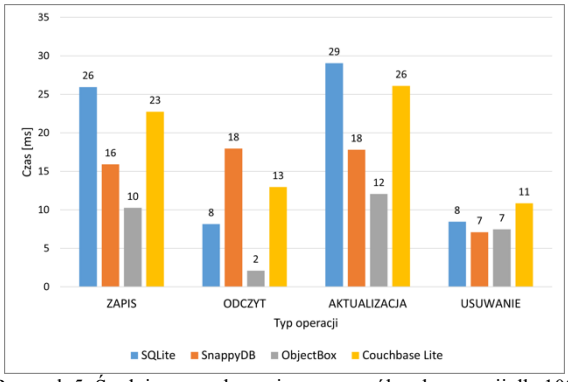
## 6. Wyniki

### 6.1. Wyniki dla scenariusza badawczego S1

Rysunki 4 oraz 5 przedstawiają rezultaty przeprowadzonego eksperymentu S1. Zauważalne jest, że najkrótsze czasy wykonywania operacji zapisu uzyskano przy użyciu bazy danych SnappyDB na urządzeniu Samsung Galaxy S10 oraz z użyciem ObjectBox na urządzeniu Samsung Galaxy S21. Warto również zauważyć, że jeśli chodzi o operacje odczytu, aktualizacji i usuwania, to w obu przypadkach najbardziej efektywna okazała się baza danych ObjectBox.



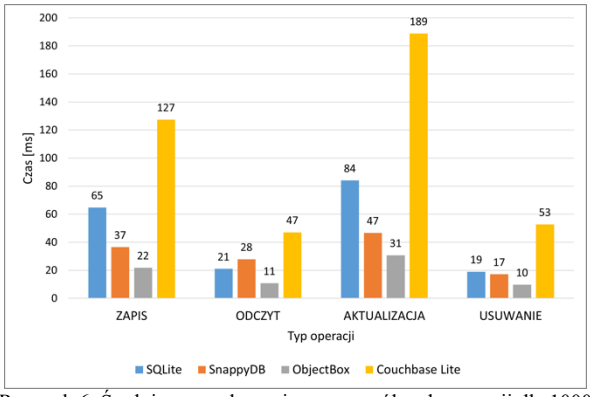

Rysunek 4: Średni czas wykonania poszczególnych operacji dla 100 rekordów i urządzenia Samsung Galaxy S10.



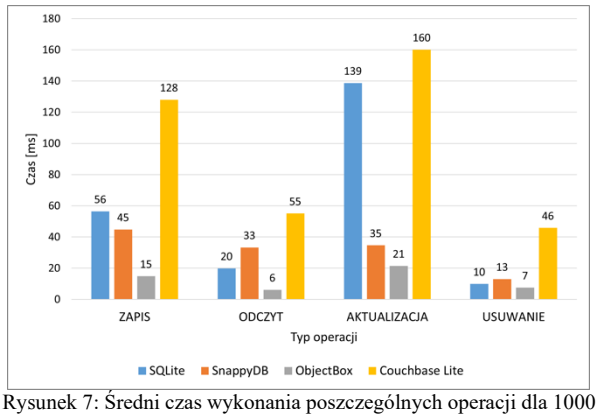

Rysunek 5: Średni czas wykonania poszczególnych operacji dla 100 rekordów i urządzenia Samsung Galaxy S21.

### 6.2. Wyniki dla scenariusza badawczego S2

Rysunki 6 oraz 7 przedstawiają wyniki scenariusza badawczego S2. W tym przypadku, baza danych ObjectBox osiągnęła zdecydowanie najlepsze wyniki na obu urządzeniach. Warto zaznaczyć, że najdłuższy czas wykonania operacji został odnotowany przy użyciu bazy danych Couchbase Lite.



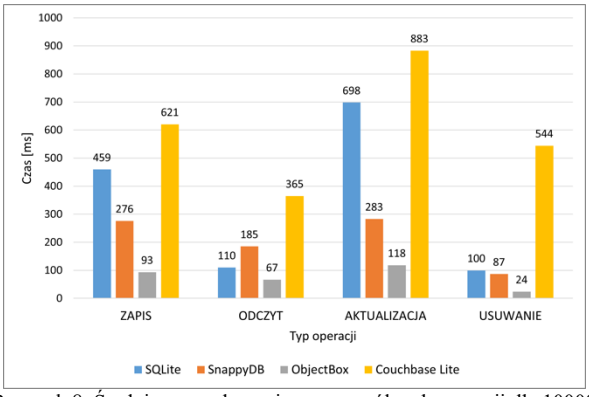

Rysunek 6: Średni czas wykonania poszczególnych operacji dla 1000 rekordów i urządzenia Samsung Galaxy S10.



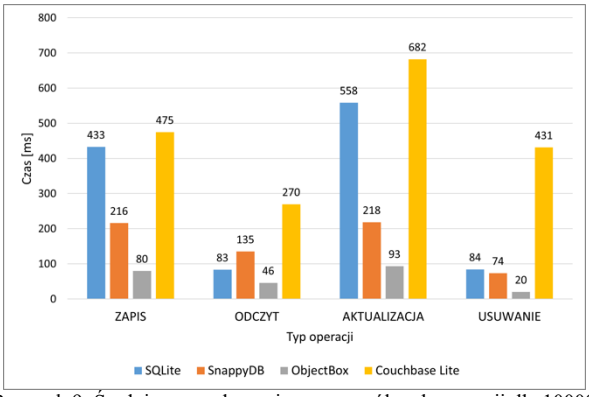

Rysunek 7: Średni czas wykonania poszczególnych operacji dla 1000 rekordów i urządzenia Samsung Galaxy S21.

### 6.3. Wyniki dla scenariusza badawczego S3

Rysunki 8 oraz 9 przedstawiają rezultaty scenariusza badawczego S3. W obu przypadkach, baza danych ObjectBox osiągnęła najlepsze wyniki. Najbardziej zbliżone do siebie wartości przedstawiały wyniki baz danych SnappyDB oraz SQLite. Warto zauważyć, że baza danych Couchbase Lite notowała najdłuższe czasy wykonania wszystkich operacji.




Rysunek 8: Średni czas wykonania poszczególnych operacji dla 10000 rekordów i urządzenia Samsung Galaxy S10.

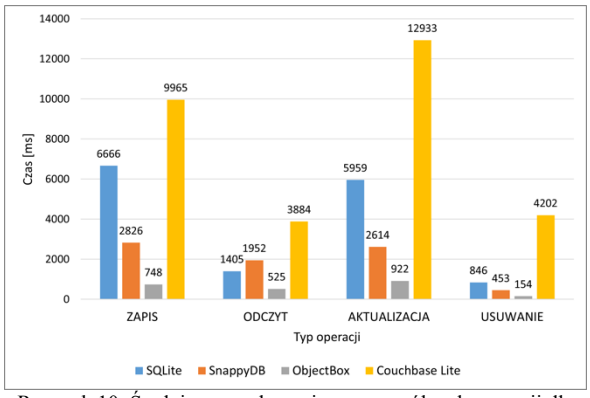



Rysunek 9: Średni czas wykonania poszczególnych operacji dla 10000 rekordów i urządzenia Samsung Galaxy S21.

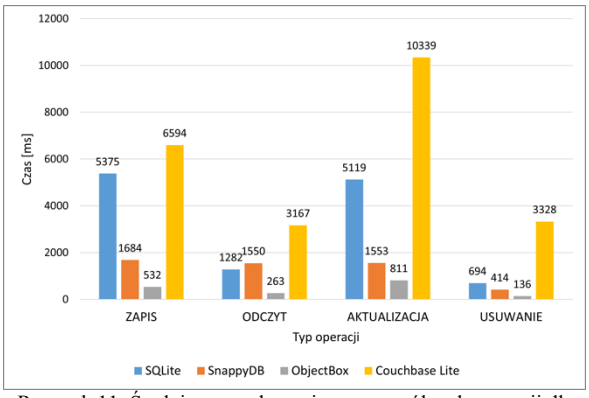
### 6.4. Wyniki dla scenariusza badawczego S4

Rysunki 10 oraz 11 przedstawiają wyniki scenariusza badawczego S4. Wszystkie operacje wykonywane przy użyciu bazy danych ObjectBox osiągały najkrótsze czasy, podczas gdy baza danych Couchbase Lite

odnotowała najdłuższe czasy wykonania. Różnice były bardzo wyraźne, ponieważ czasy wykonania operacji przy użyciu ObjectBox nie przekraczały jednej sekundy dla wszystkich operacji na obu urządzeniach. Jeżeli chodzi o bazę danych SnappyDB to wyniki nie przekraczały 3 sekund. Baza danych SQLite zaprezentowała wyniki przekraczające 5 sekund w przypadku operacji zapisu i aktualizacji danych. W przypadku bazy danych Couchbase Lite operacja aktualizacji danych przekroczyła 12 sekund na urządzeniu Samsung Galaxy S10 oraz 10 sekund na urządzeniu Samsung Galaxy S21.



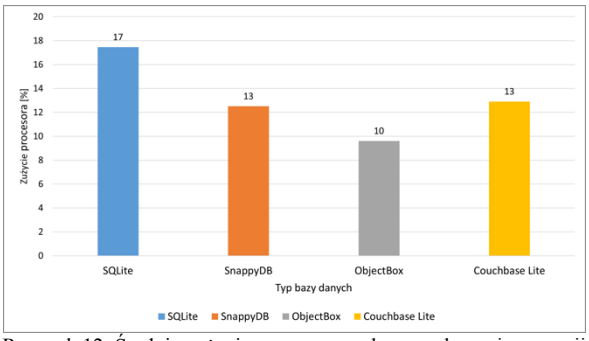

Rysunek 10: Średni czas wykonania poszczególnych operacji dla 100000 rekordów i urządzenia Samsung Galaxy S10.



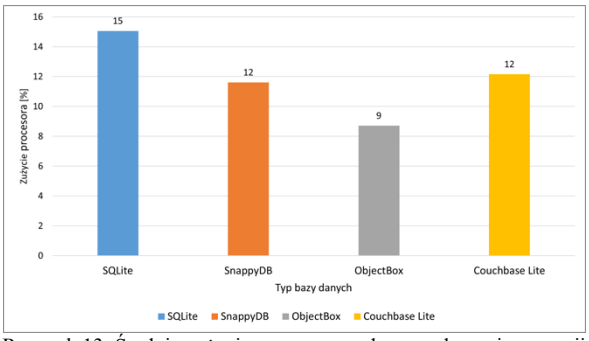

Rysunek 11: Średni czas wykonania poszczególnych operacji dla 100000 rekordów i urządzenia Samsung Galaxy S21.

### 6.5. Wyniki dla scenariusza badawczego S5

Rysunki 12 oraz 13 przedstawiają wyniki scenariusza badawczego S5. Na urządzeniu Samsung Galaxy S10 największe zużycie procesora odnotowano przy użyciu bazy danych SQLite, sięgające 17%, natomiast najbardziej efektywną bazą danych pod względem zużycia procesora okazała się ObjectBox, zużywając tylko 10% zasobów. Baza danych SnappyDB i Couchbase Lite wykazały podobne zużycie procesora, wynoszące odpowiednio 13%. Na urządzeniu Samsung Galaxy S21 zużycie procesora było podobne do wyników na Samsungu Galaxy S10. Największe zużycie, 15% odnotowano przy użyciu bazy danych SQLite, natomiast najmniejsze, tylko 9%, odnotowano dla bazy danych ObjectBox. Zużycie procesora dla SnappyDB i Couchbase Lite wyniosło odpowiednio 12%.



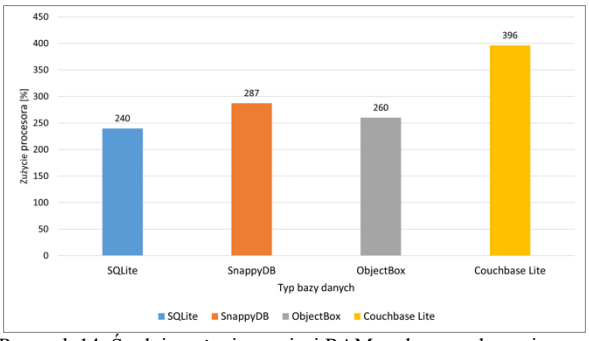

Rysunek 12: Średnie zużycie procesora podczas wykonania operacji zapisu dla 100000 rekordów i urządzenia Samsung Galaxy S10.



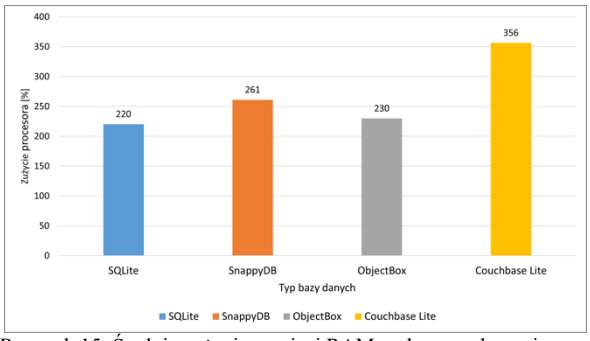

Rysunek 13: Średnie zużycie procesora podczas wykonania operacji zapisu dla 100000 rekordów i urządzenia Samsung Galaxy S21.

### 6.6. Wyniki dla scenariusza badawczego S6

Rysunki 14 oraz 15 przedstawiają wyniki scenariusza badawczego S6. Na urządzeniu Samsung Galaxy S10 największe zużycie pamięci RAM odnotowano przy użyciu bazy danych Couchbase Lite, wynoszące 396 MB. Natomiast najmniejsze zużycie pamięci zaobserwowano w przypadku SQLite, gdzie wyniosło ono 240 MB. Baza danych SnappyDB i ObjectBox wykazały podobne zużycie pamięci RAM, odpowiednio 287 MB i 260 MB. Na urządzeniu Samsung Galaxy S21 największe zużycie pamięci RAM odnotowano ponownie przy użyciu bazy danych Couchbase Lite, osiągając 356 MB. Najmniejsze zużycie pamięci zaobserwowano w przypadku SQLite, gdzie wyniosło ono 220 MB. Zużycie pamięci RAM dla SnappyDB i ObjectBox wyniosło odpowiednio 261 MB i 230 MB.




Rysunek 14: Średnie zużycie pamięci RAM podczas wykonania operacji zapisu dla 100000 rekordów i urządzenia Samsung Galaxy S10.




Rysunek 15: Średnie zużycie pamięci RAM podczas wykonania operacji zapisu dla 100000 rekordów i urządzenia Samsung Galaxy S21.

## 7. Wnioski

Na podstawie uzyskanych wyników, można wyciągnąć kilka ważnych wniosków dotyczących efektywności różnych baz danych na urządzeniach Samsung Galaxy S10 oraz Samsung Galaxy S21.

Dla mniejszych ilości danych, różnice między bazami danych są minimalne i mogą być nieodczuwalne dla użytkownika końcowego. Jednakże, wraz ze wzrostem liczby rekordów, na przykład przy operacjach na 1000, 10000 lub 100000 rekordach, różnice w czasach wykonania poszczególnych operacji stają się bardziej zauważalne. ObjectBox wykazuje się jako najbardziej efektywna baza danych dla wszystkich ilości danych, osiągając znacznie lepsze czasy wykonania operacji w porównaniu do konkurencyjnych rozwiązań. Baza danych Couchbase Lite osiągnęła zdecydowanie najgorsze czasy wykonania. Efektywność Couchbase Lite malała drastycznie wraz ze wzrostem liczby testowanych rekordów. Czasy wykonania poszczególnych operacji dla 100000 rekordów przy użyciu bazy danych Couchbase Lite wynosiły średnio od kilku do nawet kilkunastu sekund, przez co doświadczenia użytkowników mogłyby nie należeć do najlepszych.

Dodatkowo przeprowadzona analiza zużycia pamięci RAM i procesora potwierdza, że ObjectBox jest również najbardziej efektywnym rozwiązaniem pod względem wykorzystania zasobów sprzętowych na obu urządzeniach. Natomiast Couchbase Lite wykazuje największe zużycie zasobów, co może negatywnie wpłynąć na wydajność urządzenia, zwłaszcza przy operacjach na większych zbiorach danych.

Przeprowadzając analizę wyników można wywnioskować, że ObjectBox prezentuje się jako najbardziej efektywna baza danych dla wszystkich scenariuszy badawczych.

## Literatura

[1] Udział w rynku systemów operacyjnych na świecie w roku 2024, https://gs.statcounter.com/os-market-share, [12.02.2024].

[2] Statystyki użycia systemu operacyjnego Android w roku 2024, https://www.businessofapps.com/data/android-statistics, [13.02.2024].

[3] Bliższe spojrzenie na 3 najlepsze wbudowane bazy danych: SQLite, RocksDB, and DuckDB, https://hackernoon.com/a-closer-look-at-the-top-3-embedded-databases-sqlite-rocksdb-and-duckdb, [13.02.2024].

[4] Jak wybrać wbudowaną bazę danych, https://dev.to/objectbox/embedded-databases-what-is-an-embedded-database-and-how-to-choose-one-27m8, [15.02.2024].

[5] K. Wałachowski, G. Kozieł, Comparative analysis of database systems dedicated for Android, Journal of Computer Sciences Institute 15 (2020) 126–132, https://doi.org/10.35784/jcsi.2043.

[6] N. Obradovic, A. Kelec, I. Dujlovic, Performance analysis on Android SQLite database, 2019 18th International Symposium INFOTEH-JAHORINA (INFOTEH) (2019) 1-4, https://doi.org/10.1109/INFOTEH.2019.8717652.

[7] C. Ferreira, M. Lopes, L. Correia, C. Wanzeller, F. Sa, P. Martins, Database Performance on Android Devices, A Comparative Analysis, 2023 Marketing and Smart Technologies 337 (2023) 261-273, https://doi.org/10.1007/978-981-19-9099-1_18.

[8] K. Kussainov, B. Kumalakov, Mobile Data Store Platforms: Test Case based Performance Evaluation, Proceedings of the 8th International Joint Conference on Knowledge Discovery, Knowledge Engineering and Knowledge Management 0IC3K (2016) 95-99, https://doi.org/10.5220/0006032300950099.

[9] Oficjalna dokumentacja bazy danych SQLite, https://www.sqlite.org/docs.html, [17.02.2024].

[10] Oficjalna dokumentacja bazy danych SnappyDB, https://github.com/nhachicha/SnappyDB, [17.02.2024].

[11] Oficjalne repozytorium bazy danych ObjectBox, https://github.com/objectbox/objectbox-java, [17.02.2024].

[12] Oficjalna dokumentacja bazy danych Couchbase Lite, https://docs.couchbase.com/couchbase-lite/current/index.html, [17.02.2024].