JOURNAL of COMPUTER SCIENCES INSTITUTE

JCSI 7 (2018) 168-171
Wysłane: 2017-12-04
Przyjęte: 2017-12-09

Porównanie wydajności technologii Xamarin i Java przy pracy z bazą danych

Oleh Datsko*, Elżbieta Miłosz
Politechnika Lubelska, Instytut Informatyki, Nadbystrzycka 36B, 20-618 Lublin, Polska

Streszczenie. Praca z bazą danych jest jedną z podstawowych rzeczy przy tworzeniu aplikacji. Każda technologia/język oprogramowania używa własnych sposobów do pracy z bazą. W przypadku Android jest używany system zarządzania bazą danych SQLite. Przedstawiona analiza dotyczy zapisu, odczytu i usuwania danych z tabeli. Ze względu na to, że system został ograniczony w porównaniu do wersji desktopowej, dla każdego eksperymentu jest używana ograniczona liczba danych.

Słowa kluczowe: analiza; Xamarin; Java; baza danych

*Autor do korespondencji.

Adres e-mail: oleh.datsko@pollub.edu.pl

Performance comparison between Xamarin and Java database operations

Oleh Datsko*, Elżbieta Miłosz
Institute of Computer Science, Lublin University of Technology, Nadbystrzycka 36B, 20-618 Lublin, Poland

Abstract. Work with database is one of base things in developing an application. Every technology/language uses own ways to work with database. In the case of Android the SQLite relational database management system is used. Analysis applies the read, write and delete of items from the table. SQLite is like a minimized version of desktop database management system so for every experiment very limited elements count is used.

Keywords: analysis; Xamarin; Java; database

*Corresponding author.

E-mail address: oleh.datsko@pollub.edu.pl

1. Wstęp

Głównym problemem współczesnych urządzeń mobilnych jest czas ich pracy na jednym naładowaniu baterii przy dużych obciążeniach często spotykanych w aplikacjach mobilnych wykonujących różne operacje, na przykład praca z bazą danych.

Bazą danych SQLite jest plik o formacie dowolnym wybranym przez użytkownika, który zapisuje się jako zwykły plik w systemie z tym tylko, że jest w formacie binarnym, więc jest dostępny w wielu systemach operacyjnych bez dodatkowych przetwarzań [1]. Użycie SQLite w Xamarin oraz Java prawie się nie różni, dlatego implementacja jest prawie taką samą, a składnia SQLite jest niezmienną na wszystkich platformach bądź to Android czy Windows). Obie technologie są bardzo podobne nawet w tym, że umieszczają klasy SQLite w tym samym miejscu – w pakiecie android.dabatase.sqlite w Javie [2] oraz w przestrzeni nazw Android.Database.Sqlite w Xamarin [3].

Wszystkie dane można przechowywać na dysku w zwykłym pliku, odczytując i zapisując go za każdym razem przy zmianie danych, ale są przypadki, kiedy system zarządzania bazą danych ma większą kontrolę nad danymi i traci mniej czasu na akcje zapisu/odczytu/usunięcia. Zgodnie z oficjalnym testowaniem, odczytywanie w SQLite jest w 2 razy szybsze, a zapisywanie jest wolniejsze niż gdyby użyto zwykłego pliku [4].

Celem artykułu jest porównanie wydajności przy pracy z systemem zarządzania bazą danych SQLite przez technologię Xamarin i Java. W zakres badania wchodzi porównanie szybkości odczytywania, zapisywania oraz usunięcia danych. Przy tym, usunięcie i odczytywanie sprawdzane jest na 2 sposoby – po kolei, oraz wszystkie za jednem razem. Różnicą tych dwóch metod jest to, że przy pierwszym nie jest używany SQLite na wszystkie 100% jak w drugim przypadku, zarządzanie wykonywaniem wszystkich operacji poszukiwania i usunięcia elementu leży całkowicie na odpowiedzialności samego systemu zarządzania bazą.

2. Plan badań

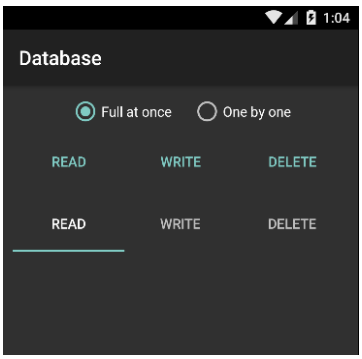
Celem badań jest sprawdzenie czasu pracy SQLite przy użyciu dwóch technologii: Xamarin.Android i Java. Dane technologie są zrealizowane w bardzo podobny sposób z tą różnicą, że są napisane używając, gdzie była taka potrzeba, wszystkich dostępnych podejść z języków programowania (C# i Java).

Hipotezy badawcze zdefiniowane do weryfikacji są następujące:

- Odczytywanie z bazy odbywa się szybciej w Java.
- Zapisywanie do bazy jest szybsze w Java.
- Usunięcie z bazy jest szybsze w Java.

Podstawą do przypuszczenia, że wyniki Javy we wszystkich trzech przypadkach będą lepsze jest to, że Android został napisany z myślą o użyciu Java (jej składni) jako technologii do tworzenia aplikacji mobilnych, a Xamarin powstał znacznie później i używa JNI (ang. Java Native Interface) i wiązanie z Java z obiektami .NET, jako pośrednik między Androidem a .NET (bezpośredni dostęp do API Androida jest niemożliwy, bo system na to nie pozwala) [5].

Dla przeprowadzenia badania zostały napisane identyczne aplikacje (Rys.1) oddzielnie w Xamarin.Android i Java, w których interfejs jest taki sam (cecha Xamarin.Android, że interfejs nie różni się od interfejsu napisanego w Java).



Rys. 1. Widok interfejsu do testowania bazy danych.

Dla odczytywania i usunięcia danych zostały wykorzystano dwie metody: pojedynczo czy wszystko naraz. Działa to w taki sposób, że wybierając wszystko naraz, baza danych zwróci lub usunie wszystkie elementy wykorzystując tylko jedną iterację w metodzie głównej, która wywołuje metodę z dostępem do bazy, a w tej już zwraca wszystkie elementy z bazy iterując używając kursora. Drugi sposób działa w taki sposób, że iteracja odbywa się w metodzie głównej, gdzie z metody dostępu do bazy zwraca się obiekt zawierający id. Krok iteracji kończy się i zaczyna się kolejny, w którym są wyszukiwane elementy z identyfikatorem id, który jest większy od przekazanego w parametrze.

Inną jest metoda zapisywania danych, ponieważ dostęp do bazy możliwy jest tylko w sposób pojedynczy, ponieważ jest konieczne przetwarzanie obiektu w inny typ, a w danym przypadku to jest ContentValues, który później przekazuje się do metody wstawienia wpisu do bazy.

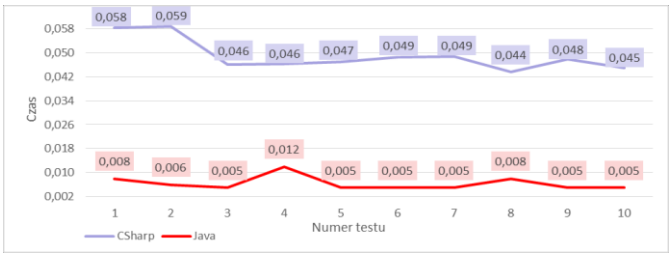
Opisane powyżej metody badania są aktualne dla Xamarin.Android i Java z tym, że w .NET używa się klasy stopwotcher dla uzyskania czasu pracy, który jest polecany zamiast DateTime, ponieważ ma większą dokładność czasu. W aplikacji Java czas pracy jest pobierany przez pakiet System, który zawiera metodę nanoTime. Nie patrząc na to, że dokładność jest bardzo duża, ale jednostką miary czasu jest milisekunda (dokładność jest do jednej milisekundy), przez co została napisana metoda przetwarzająca (metoda dodana do klasy bazowej, która dziedziczy po klasie Activity, od której dziedziczy aktywność testu).

Oprócz zapisywania czasu w samej aplikacji, czas wykonywania metody tak samo jest pobierany z Android Monitor – potężnego narzędzia dla debugowania Android aplikacji, które zawiera wykresy używania CPU, pamięci operacyjnej, sieci oraz GPU aplikacji [6]. Także narzędzie pozwala na logowanie akcji z kodu za pośrednictwem narzędzia z nazwą logcat.

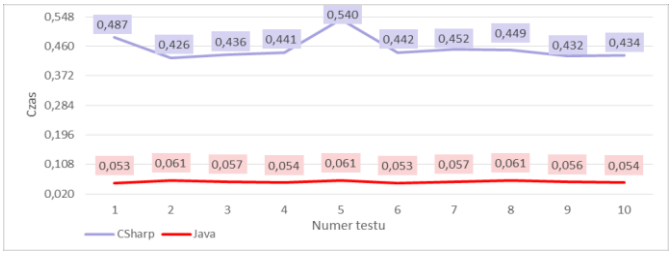
Przeprowadzone badanie zawiera wyniki zapisywania 10 testów (na każdą z dwóch metod). Przed każdym eksperymentem aplikacja uruchamia się ponownie, żeby oczyścić śmieci, które pozostały od poprzedniego eksperymentu (teoretycznie nie jest to potrzebne, ale dane mogą mieć niegatywny wpływ na kolejny test). Eksperyment z odczytywaniem bazy danych (za jednym razem) przeprowadzono dla 100, 1000, 10000 i 100000 elementów, dla zapisywania i usunięcia za jednym razem eksperyment przeprowadzono dla 100, 1000 oraz 10000 rekordów. Dla usunięcia przeprowadzono jeszcze kolejny test z usunięciem pojedynczym, który zawiera 100 elementów.

3. Wyniki badań

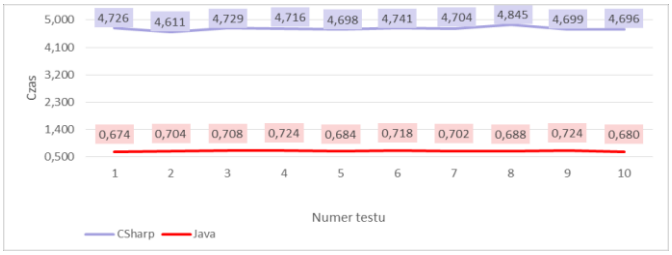
Wyniki badań są przedstawione w postaci wykresów (Rys.2 - Rys.12) oraz w tabeli porównawczej (Tabela 1), gdzie znajdują się procentowe wartości czasu wykonywania i obciążenia CPU. Czas podany na wykresach jest w sekundach.



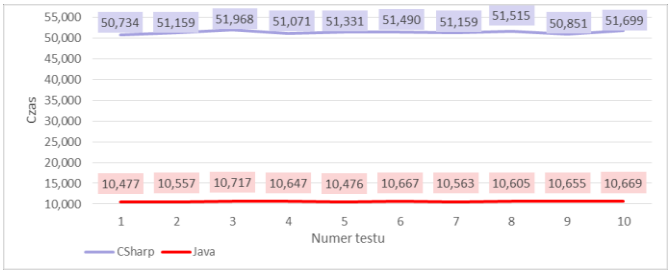
Rys. 2. Test 100 elementów przy odczytywaniu bazy danych.



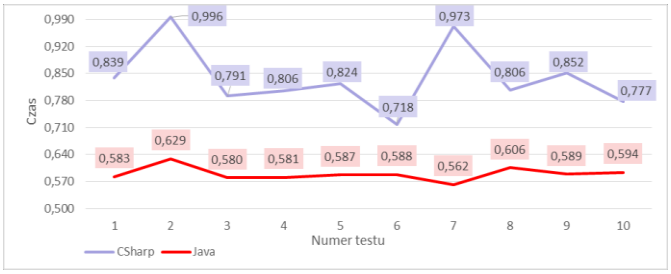
Rys. 3. Test 1000 elementów przy odczytywaniu bazy danych.



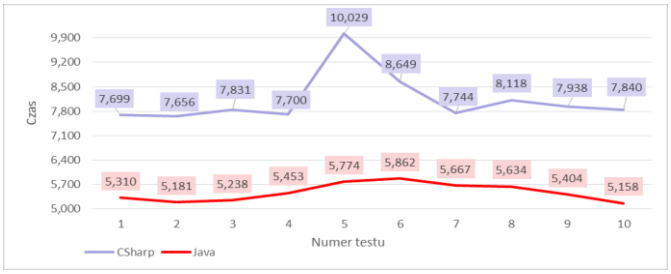
Rys. 4. Test 10000 elementów przy odczytywaniu bazy danych.



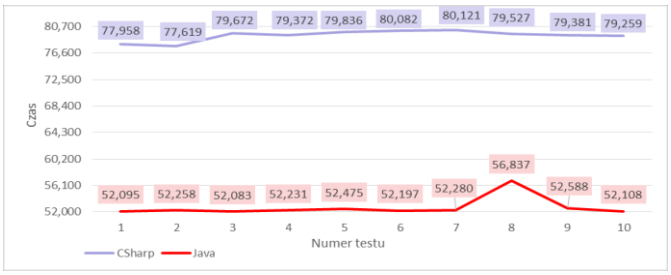
Rys. 5. Test 100000 elementów przy odczytywaniu bazy danych.



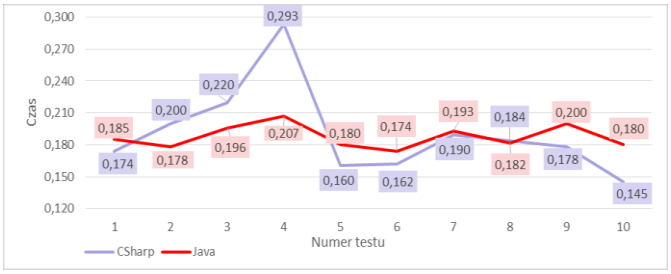
Rys. 6. Test 100 elementów przy zapisie do bazy danych.



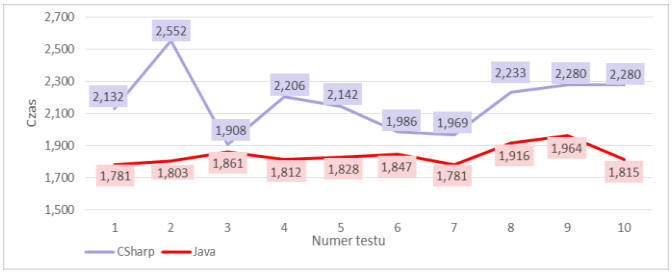
Rys. 7. Test 1000 elementów przy zapisie do bazy danych.



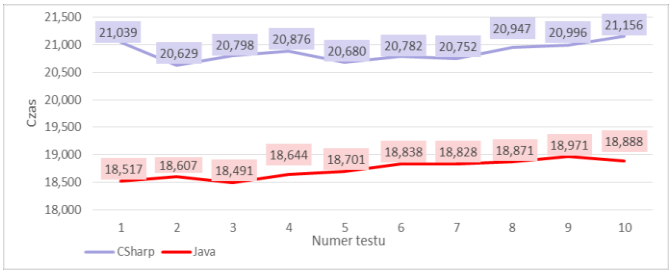
Rys. 8. Test 10000 elementów przy zapisie do bazy danych.



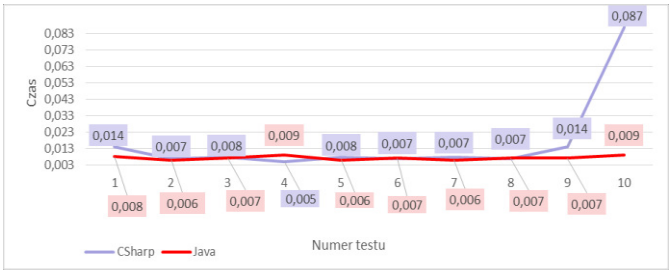
Rys. 9. Test 100 elementów przy usunięciu z bazy danych.



Rys. 10. Test 1000 elementów przy usunięciu z bazy danych.



Rys. 11. Test 10000 elementów przy usunięciu z bazy danych.



Rys. 12. Test 100 elementów przy pojedynczym usunięciu z bazy danych.

4. Wnioski

Na podstawie przeprowadzonych badań otrzymano wyniki szybkości wykonywania operacji z bazą danych używając technologii Xamarin.Android i Java (Tabele 1-3).

Dodatkowy test został przeprowadzony dla pojedynczego usunięcia z tabeli 100 elementów. Wyniki testu są następujące: czas usunięcia wykorzystując Java jest na 125,6% mniejszy niż w Xamarin.Android. Przy tym, obciążenie CPU jest na 37,3% mniejsze na korzyść Java.

Tabela. 1. Dane (w procentach) Java w porównaniu do Xamarin.Android przy odczytaniu danych (minus oznacza przewagę Xamarin.Android)

Liczba elementów	Czas	CPU (aplikacja)	CPU (jądro)
100	665,3	2,3	13,0
1000	700,3	45,9	10,3
10000	573,2	-3,9	-71,4
100000	383,8	-0,9	-28,6

Z Tabeli 1 wynika, że odczyt danych przy użyciu Java jest od 3 do 7 razy szybszy niż analogiczny, obciążenie CPU aplikacją jest mniejsze przy mniejszej ilości danych (od 2,3% do 45,9%), ale przy dużej ilości danych zwycięża Xamarin.Android.

Tabela. 2. Dane (w procentach) Java w porównaniu do Xamarin.Android przy zapisywaniu danych (minus oznacza przewagę Xamarin.Android)

Liczba elementów	Czas	CPU (aplikacja)	CPU (jądro)
100	42,1	32,2	8,7
1000	48,5	7,8	-39,6
10000	50,4	37,0	-27,8

Z Tabeli 2 wynika, że szybkość zapisu do bazy nie jest na takim samym poziomie zwycięstwa jak to było przy odczycie, ale czas wykonywania tej samej operacji jest na 40%-50% mniejszy od analogicznego w Xamarin.Android. Jeżeli chodzi o obciążenie CPU to Java tu tak samo pokazuje absolutne zwycięstwo.

Zgodnie z Tabelą 3, czas wykonywania w Java jest mniejszy na 1%-18% niż analogiczny w Xamarin.Android. Z tym, że w teście z 1000 elementów, obciążenie CPU jest na 23,5% mniejsze w Xamarin.Android.

Tabela. 3. Dane (w procentach) Java w porównaniu do Xamarin.Android przy pojedynczym usunięciu danych (minus oznacza przewagę Xamarin.Android)

Liczba elementów	Czas	CPU (aplikacja)	CPU (jądro)
100	1,6	39,6	31,3
1000	17,8	-23,5	-39,6
10000	11,4	16,7	-46,2

Wyniki przedstawione w tabelach 1-3 potwierdzają założone hipotezy:

1. Odczytywanie z bazy odbywa się szybciej w Java.
2. Zapisywanie do bazy jest szybsze w Java.
3. Usunięcie z bazy jest szybsze w Java.

Literatura

[1] G. Allen, M. Owens, The Definitive Guide to SQLite, Apress, 2010.

[2] android.database.sqlite, https://developer.android.com/reference/android/database/sqlite/package-summary.html [01.11.2017].

[3] Android.Database.Sqlite Namespace, https://developer.xamarin.com/api/namespace/Android.Database.Sqlite/ [01.11.2017]

[4] 35% Faster Than The Filesystem, https://www.sqlite.org/fasterthanfs.html [04.11.2017]

[5] Working With JNI, https://developer.xamarin.com/guides/android/advanced_topics/java_integration_overview/working_with_jni/ [05.11.2017]

[6] Android Monitor Basis https://developer.android.com/studio/profile/am-basics.html#byb [05.11.2017]