JCSI 2 (2016) 133-139
Journal of Computer Sciences Institute

Wysłane: 2016-11-21
Przyjęte: 2016-11-30

Porównanie możliwości wykorzystania oraz analiza wydajności baz danych na systemach mobilnych

Mateusz Grudzień*, Konrad Korgol*, Dariusz Gutek
Politechnika Lubelska, Instytut Informatyki, Nadbystrzycka 36B, 20-618 Lublin, Polska

Streszczenie. Artykuł przedstawia wybrane formy składowania danych tj. Local Storage, Shared Preferences, pliki płaskie oraz SQLite w kontekście dwóch systemów mobilnych – Android oraz Windows Mobile. Opisuje on również sposoby, poprzez które możliwe jest połączenie z zewnętrznymi systemami bazodanowymi takimi jak Microsoft SQL Server, PostgreSQL czy MySQL i stara się odpowiedzieć na pytanie, użycie której z tych opcji ma największy sens w poszczególnych przypadkach. Całość argumentuje badaniami wydajności, którym poddane zostały wszystkie z wymienionych opcji.

Słowa kluczowe: Android; Windows Mobile; baza danych; urządzenie mobilne

*Autorzy do korespondencji.

Adresy e-mail: mateusz.grudzien@protonmail.com, konrad.korgol@gmail.com

Comparison of the possible uses and performance analysis of databases on mobile operating systems

Mateusz Grudzień*, Konrad Korgol*, Dariusz Gutek
Politechnika Lubelska, Instytut Informatyki, Nadbystrzycka 36B, 20-618 Lublin, Polska

Abstract. This publication presents chosen forms of data persistence such as: Local Storage, Shared Preferences, flat files and SQLite in the context of two widely used mobile operating systems – Android and Windows Mobile. It also describes ways to connect to external database engines such as Microsoft SQL Server, PostgreSQL or MySQL and tries to answer the question which one of these data persistence forms makes the most sense and when. The arguments are based on performance tests that all of the described solutions were participants of.

Keywords: Android; Windows Mobile; database; mobile device

*Corresponding authors.

E-mail addresses: mateusz.grudzien@protonmail.com, konrad.korgol@gmail.com

1. Wstęp

Obecnie obserwowane tempo rozwoju technologii niejednego świadomego obserwatora przyprawić może o zawrót głowy. Istnieje wiele przykładów gdzie to, co w połowie ubiegłego stulecia składało się na dobry film science-fiction, dziś dla większości ludzi jest częścią ich codziennego życia. Tak szybki postęp dokonuje się praktycznie we wszystkich gałęziach świata technologii a niejednokrotnie rozwój w jednej dziedzinie jest zalążkiem dla powstania całkowicie nowej gałęzi. Sytuacja taka miała miejsce w przypadku postępującej miniaturyzacji, która najpierw dała narodziny przenośnym telefonom komórkowym, które kilkanaście lat później zostały z kolei zastąpione przez tzw. smartfony. Urządzenie, które prawie każdy z nas nosi dziś w kieszeni, za pomocą internetu i szybkiej transmisji danych, daje swojemu posiadaczowi możliwość kontaktu i wymiany wiedzy oraz poglądów z osobami znajdującymi się na drugim końcu globu, wykonanie i przesłanie wysokiej jakości zdjęcia czy nagrania wideo. Przede wszystkich jednak daje ono dostęp do ogromnych zasobów informacji, które trzeba nie tylko składować, ale również odpowiednio przetworzyć.

Jednym ze skutków spowodowanych rozwojem cyfrowego świata jest właśnie rosnąca ilość wytwarzanych danych. W 2010 roku, zarządzający wówczas Google, Eric Schmidt ogłosił że wg szacunków prowadzonych przez kierowaną przez niego firmę, od zarania dziejów do roku 2003, ludzkość wytworzyła 5 eksabajtów danych. W momencie, w którym Schmidt podzielił się tym wynikiem – w roku 2010 – czas potrzebny naszej cywilizacji na wytworzenie tej samej ilości danych wynosił już jedynie 2 dni. Chociaż od tamtego czasu nikt nie pokusił się o odnowienie tych szacunków, spodziewać się można, że ilość danych jakie przetwarzamy jedynie wzrosła, a trend wydaje się wcale nie wykazywać oznak spowolnienia.

Zestawiając ze sobą oba te fakty, tj. rozwój technologii mobilnych oraz gwałtownie rosnącą ilość wytwarzanych danych, pytanie jakie automatycznie nasuwa się na myśl brzmi: jak dobrze dostosowane do przetwarzania takiej ilości danych są dostępne obecnie urządzenia mobilne. Czy sposoby składowania danych dostępne na nowoczesnych mobilnych systemach operacyjnych oferują przystępne czasy dostępu do danych?

2. Opis obiektów badań

Systemy mobilne oferują mnogą liczbę sposobów przechowywania danych. SQLite, pliki płaskie czy zewnętrzne bazy danych są to formy występujące na wielu systemach mobilnych. Istnieją też takie formy składowania danych, które są dostępne tylko i wyłącznie na konkretnym systemie. Mowa tutaj o Shared Preferences, które obecne jest na Androidzie,

a także Local Settings istniejące na platformie Windows Mobile. Co prawda docelowy system obu form się różni, jednakże sposób, w jaki przechowują one dane, jest jednakowy i jest to para klucz-wartość[4]. Przechowywanie oraz dostęp do samych danych za pomocą tych form jest niewątpliwie najprostszy. Takie rozwiązanie świetnie sprawdzi się w przypadku przechowywania ustawień aplikacji czy kont użytkownika.

W przypadku, gdy aplikacja pobiera dużo danych z internetu, te w postaci zdjęć czy filmów warto umieszczać jako pliki płaskie w lokalnej pamięci urządzenia[3]. Dodatkowo w systemie Android takie pliki mogą być przechowywane w tzw. Internal Storage[14] oraz External Storage[13]. Rozwiązanie to zapewnia oszczędność transferu danych użytkownika, gdyż aplikacja, z której korzysta użytkownik, nie pobiera danych ponownie, lecz odwołuje się do tych wcześniej zapisanych. O ile przechowywanie takich danych w postaci plików płaskich nosi ze sobą wiele zalet, o tyle przechowywanie danych użytkowników w postaci rekordów danych nie ma większego sensu. Niesie to ze sobą wysokie koszty czasowe dostępu do danych oraz mało efektywne zarządzanie samymi danymi.

W tej sytuacji najlepszym rozwiązaniem jest lokalna baza danych SQLite. Przechowuje ona w postaci binarnej ustrukturyzowane dane aplikacji, które umieszczone są w tabelach i mogą posiadać między sobą relacje[2]. Do bazy dostęp ma tylko i wyłącznie aplikacja dla której baza została stworzona, zaś sam użytkownik może dotrzeć do bazy SQLite tylko i wyłącznie posiadając prawa administratora systemu[6].

W dobie wszechobecnego szybkiego dostępu do internetu należy zwrócić uwagę na bazy danych dostępne na zewnętrznych serwerach. MySQL, PostgreSQL czy Microsoft SQL Server to tylko kilka przykładów baz danych, które mogą zostać wdrożone na takim serwerze. Aplikacje korzystające z takiej formy składowania danych oferują użytkownikom dostęp do danych nawet w sytuacji, gdy urządzenie użytkownika zostanie zniszczone lub zostanie wykonana operacja czyszczenia danych w całym systemie. Wystarczy, że użytkownik zaloguje się na jego indywidualne konto w aplikacji i będzie on posiadał już wszystkie wprowadzone dane. Należy tutaj zwrócić uwagę na fakt, że o ile takie rozwiązanie niesie ze sobą tę nieocenioną zaletę, jak nieulotność danych, tak też posiada ono bardzo ważną wadę. W przypadku braku dostępu do internetu o korzystaniu z aplikacji można zapomnieć, a samo utrzymanie serwera ponosi ze sobą rozmaite koszty.

Każda przedstawiona forma przechowywania danych może być łatwo i skutecznie zaimplementowana w aplikacji, dzięki bogatej bibliotece odpowiednich klas w pakiecie SDK dla danego systemu mobilnego. W przypadku systemu Android i Shared Preferences jest to klasa SharedPreferences[10], dla SQLite są to klasy SQLiteOpenHelper[12], SQLiteDatabase[11], ContentValues oraz klasa Cursor[7]. Operacje na plikach mogą zostać łatwo wykonane poprzez klasy File, FileOutputStream[8], a także FileInputStream[9]. Zewnętrzne bazy danych najlepiej obsługiwać poprzez REST API znajdujące się na serwerze. Powinny być one również obsługiwane asynchroniczne dzięki czemu operacja pobierania i zapisu danych odbywa się w wątku działającym w tle, co eliminuje negatywny efekt braku odpowiedzi aplikacji na komendy użytkownika przez określoną ilość czasu potrzebną na wykonanie operacji. Komunikaty przychodzące oraz zwrotne warto parsować do popularnego formatu JSON, który zapewnia standaryzację komunikacji oraz łatwość manipulacji danymi[1]. Nie należy również zapominać o pisaniu optymalnego kodu aplikacji oraz optymalizacji zapytań do bazy danych[5].

3. Metoda badań

Badaniu poddane zostały wybrane formy przechowywania danych na dwóch mobilnych systemach operacyjnych. Opcja unikalna dla systemu Windows Mobile to Local Storage, które udostępnia API umożliwiające dostęp do pamięci wewnętrznej urządzenia oraz do kontenera Local Settings przechowującego pary klucz-wartość. Opcje unikalne dla systemu Android to odpowiednik Local Settings z systemu Windows Mobile czyli Shared Preferences oraz Internal i External Storage. Opcja poniekąd wspólna, poniekąd, bo na obu systemach różniąca się implementacją, to wewnętrzna baza danych SQLite. Wreszcie, na obu systemach, badaniu poddane zostały również zewnętrzne bazy danych takie jak Microsoft SQL Server, PostgreSQL oraz MySQL. Dostęp do tych silników bazodanowych zrealizowany został poprzez przygotowany w tym celu serwer PHP, który działał jako pośrednik wystawiając przygotowane na potrzeby badań REST API. Urządzenia mobilne łączyły się z serwerem poprzez lokalną sieć WiFi.

Wersje wykorzystanego oprogramowania to:
- Android 5.0.1,
- Windows Mobile 10.0.14393.448,
- SQLite dla WM10: v. 3.15.1,
- SQLite dla Androida: 3.8.4.3,
- Microsoft SQL Server 2016,
- PostgreSQL 9.3,
- MySQL 5.7.14.

Badania dla systemu Windows Mobile wykonywane były na urządzeniu Microsoft Lumia 640XL o specyfikacji:
- CPU: Qualcomm MSM8226 Snapdragon 400, 4x1.2 GHz Cortex-A7,
- pamięć: 8GB (brak informacji o szybkości pamięci),
- WiFi 802.11 b/g/n.

Badania dla systemu Android przeprowadzane były na telefonie Samsung Galaxy I9505. Specyfikacja urządzenia:
- CPU: Qualcomm Snapdragon 600, 4x1,9GHz,
- pamięć: 16GB, zapis: 22.95MB/s, odczyt 92.05MB/s,
- WiFi v802.11 a/b/g/n/ac 2,4GHz i 5GHz.

Serwer, na którym uruchomione były baz danych oraz który oferował dostęp do nich poprzez REST API:
- CPU: Intel Core I7-4800MQ 2.70GHz,
- pamięć: Kingston HyperX 16GB DDR3 PC1300,
- dysk twardy: SSD Kingston V300 R/W 450MB/s.

Zarówno w przypadku systemu Windows Mobile jak i Android przygotowane zostały specjalne aplikacje, które automatyzowały proces badawczy. Na podstawie danych wprowadzonych przez prowadzącego badania, takich jak: rozmiar danych, liczba rekordów, liczba powtórzeń badania

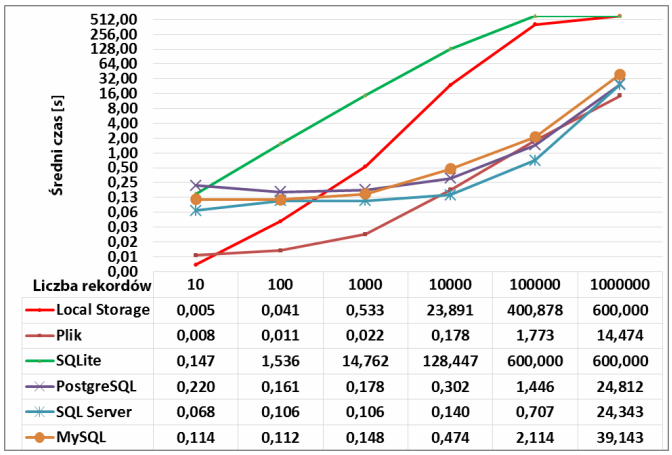
oraz wybrana opcja składowania danych do badania, aplikacja, w przypadku badania czasów zapisu, generowała zadaną ilość losowych danych, a następnie wykonywała żądanie zapisu. Czas w tym przypadku mierzony był od momentu wysłania żądania zapisu wygenerowanych już danych do momentu otrzymania odpowiedzi o sukcesie operacji. W przypadku żądania odczytu danych aplikacja liczyła czas operacji od momentu wysłania żądania o odczyt danych do momentu, gdy otrzymane dane zostały przetworzone do takiej samej formy w jakiej zostały one odesłane do zapisu. Jest to o tyle ważne, że np. w przypadku odczytu rekordów z danymi tekstowymi z pliku płaskiego, zawartość pliku musiała zostać poddana odpowiedniemu parsowaniu. Pomiar czasu w systemie Windows Mobile stosowany był poprzez klasę Stopwatch z pakietu System.Diagnostics, w systemie Android natomiast za pośrednictwem klasy Date i jej metody getTime().

Typy danych, które poddane zostały badaniu to: pary klucz-wartość, rekordy z danymi tekstowymi o długości 255 znaków, 100 znaków oraz 10 znaków, rekordy ze zmiennoprzecinkowymi danymi numerycznymi zapisywanymi na 32-bitach, rekordy z danymi binarnymi o rozmiarze 10kB, 1kB i 100B oraz rekordy zawierające różne typy danych. Rekord z typami mieszanymi składał się z liczby całkowitej zapisanej na 32 bitach, liczby zmiennoprzecinkowej o takim samym rozmiarze, daty, losowego ciągu o długości 255 znaków oraz danych binarnych o rozmiarze 100 bajtów.

4. Wyniki badań

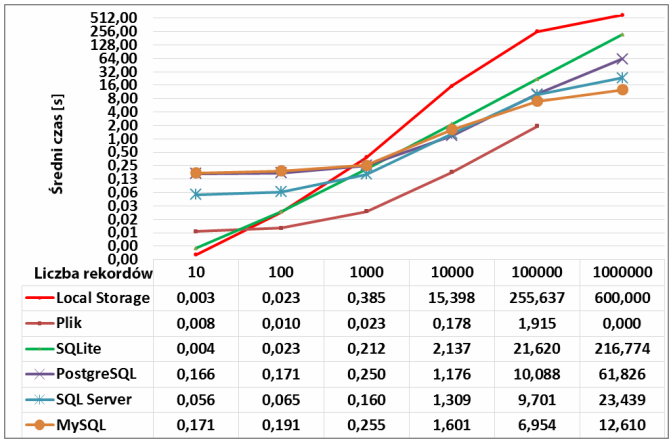
Poniżej przedstawione zostaną wyniki części przeprowadzonych badań. Wykresy widoczne na rys. 1 do rys. 20 przedstawiają wyniki kolejno badań zapisu i odczytu par klucz-wartość, rekordów zawierających dane tekstowe, rekordów zawierających dane będące liczbami zmiennoprzecinkowymi, rekordów zawierających dane binarne oraz rekordów, na które składały się dane o różnych typach. Dla każdej z tych opcji przedstawione zostały wyniki najpierw dla systemu Windows Mobile, a następnie Android. Do każdego wykresu dołączona została legenda, na której widoczne są osiągnięte średnie czasy dla każdej uczestniczącej w tym konkretnym badaniu opcji składowania danych. Dla przykładu: na rys. 1, na przecięciu kolumny oznaczonej wartością 1000 oraz wiersza oznaczonego wartością SQLite, odczytać można liczbę 14,762. Oznacza to, że zapis 1000 par klucz-wartość do bazy danych SQLite na systemie Windows Mobile zajął średnio 14,762 sekundy. Wartości te naniesione zostały na znajdujące się wyżej wykresy, gdzie wartości na pionowej osi oznaczają średni czas w sekundach, jaki upłynął do ukończenia badania. Na osi poziomej z kolei widoczna jest liczba rekordów, na których operowało to badanie lub w przypadku wykresów z rys.1 do rys. 4 jest to liczba zapisywanych bądź odczytywanych par klucz-wartość. W przypadku, gdy z jakiegoś powodu w danym scenariuszu badanie nie zostało ukończone (np. urządzenie, na którym przeprowadzano badanie posiadało zbyt małą ilość pamięci), wynik dla takiego przypadku oznaczany był jako 0 a linia wyników nie uwzględnia go na wykresie.

Rysunek 1 oraz rysunek 2 przedstawiają wyniki badań zapisu oraz odczytu par klucz-wartość na systemie Windows Mobile.



Liczba rekordów	10	100	1000	10000	100000	1000000
Local Storage	0,005	0,041	0,533	23,891	400,878	600,000
Plik	0,008	0,011	0,022	0,178	1,773	14,474
SQLite	0,147	1,536	14,762	128,447	600,000	600,000
PostgreSQL	0,220	0,161	0,178	0,302	1,446	24,812
SQL Server	0,068	0,106	0,106	0,140	0,707	24,343
MySQL	0,114	0,112	0,148	0,474	2,114	39,143

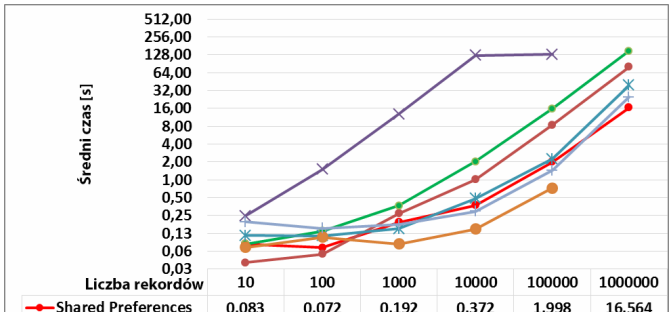
Rys. 1. Wyniki badań czasów zapisu par klucz-wartość na systemie Windows Mobile



Liczba rekordów	10	100	1000	10000	100000	1000000
Local Storage	0,003	0,023	0,385	15,398	255,637	600,000
Plik	0,008	0,010	0,023	0,178	1,915	0,000
SQLite	0,004	0,023	0,212	2,137	21,620	216,774
PostgreSQL	0,166	0,171	0,250	1,176	10,088	61,826
SQL Server	0,056	0,065	0,160	1,309	9,701	23,439
MySQL	0,171	0,191	0,255	1,601	6,954	12,610

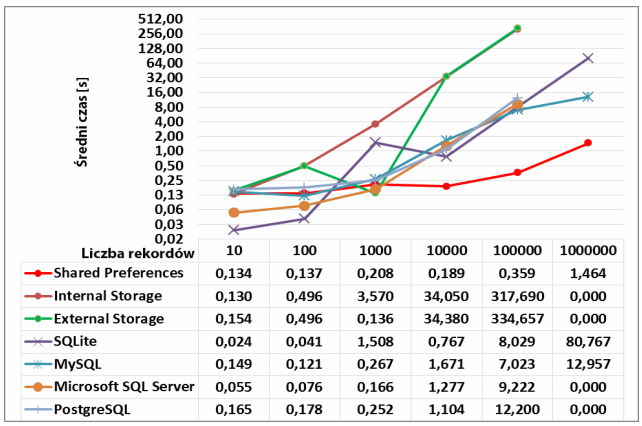
Rys. 2. Wyniki badań czasów odczytu par klucz-wartość na systemie Windows Mobile

Rysunek 3 i rysunek 4 przedstawiają wyniki dla tego samego badania, ale przeprowadzonego na systemie Android.



Liczba rekordów	10	100	1000	10000	100000	1000000
Shared Preferences	0,083	0,072	0,192	0,372	1,998	16,564
Internal Storage	0,040	0,056	0,274	1,011	8,426	80,325
External Storage	0,083	0,136	0,369	2,030	15,853	148,840
SQLite	0,248	1,508	12,893	125,018	129,736	0,000
MySQL	0,115	0,115	0,151	0,484	2,253	39,428
Microsoft SQL Server	0,073	0,109	0,084	0,148	0,728	0,000
PostgreSQL	0,196	0,152	0,175	0,296	1,444	24,990

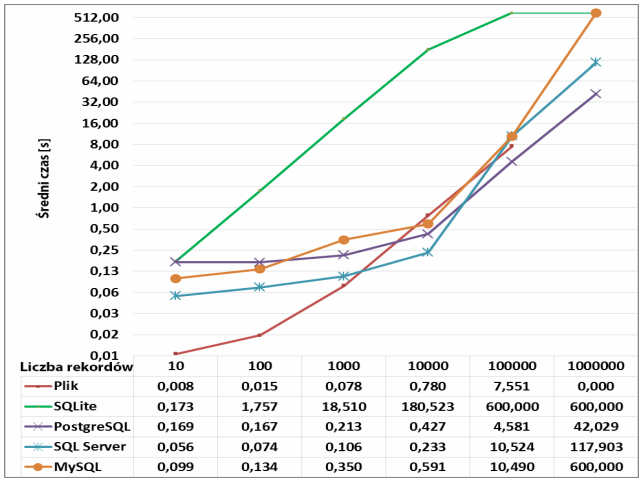
Rys. 3. Wyniki badań czasów zapisu par klucz-wartość na systemie Android



Liczba rekordów	10	100	1000	10000	100000	1000000
Shared Preferences	0,134	0,137	0,208	0,189	0,359	1,464
Internal Storage	0,130	0,496	3,570	34,050	317,690	0,000
External Storage	0,154	0,496	0,136	34,380	334,657	0,000
SQLite	0,024	0,041	1,508	0,767	8,029	80,767
MySQL	0,149	0,121	0,267	1,671	7,023	12,957
Microsoft SQL Server	0,055	0,076	0,166	1,277	9,222	0,000
PostgreSQL	0,165	0,178	0,252	1,104	12,200	0,000

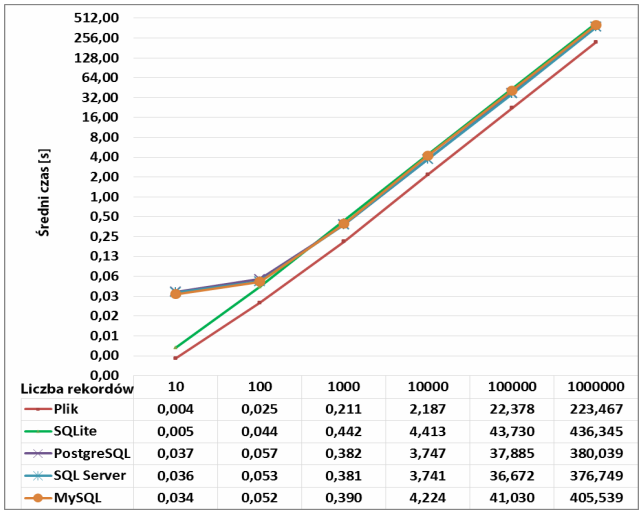
Rys. 4. Wyniki badań czasów odczytu par klucz-wartość na systemie Android

Rysunek 5 oraz rysunek 6 widoczne poniżej przedstawiają czasy jakie osiągnęły poszczególne obiekty badawcze w przypadku rekordów z danymi tekstowymi na systemie Windows Mobile.



Liczba rekordów	10	100	1000	10000	100000	1000000
Plik	0,008	0,015	0,078	0,780	7,551	0,000
SQLite	0,173	1,757	18,510	180,523	600,000	600,000
PostgreSQL	0,169	0,167	0,213	0,427	4,581	42,029
SQL Server	0,056	0,074	0,106	0,233	10,524	117,903
MySQL	0,099	0,134	0,350	0,591	10,490	600,000

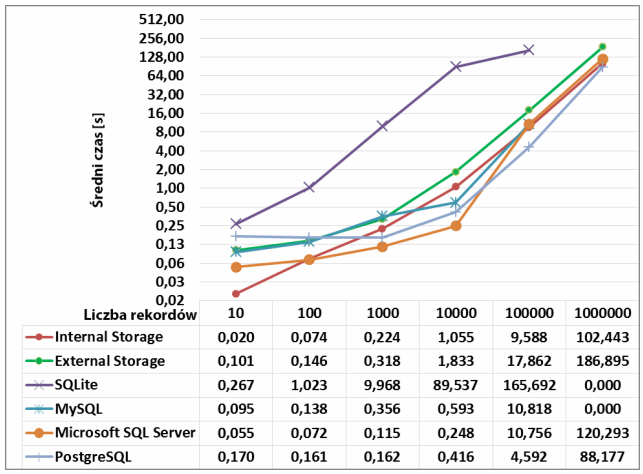
Rys. 5. Wyniki badań czasów zapisu rekordów z danymi tekstowymi na systemie Windows Mobile



Liczba rekordów	10	100	1000	10000	100000	1000000
Plik	0,004	0,025	0,211	2,187	22,378	223,467
SQLite	0,005	0,044	0,442	4,413	43,730	436,345
PostgreSQL	0,037	0,057	0,382	3,747	37,885	380,039
SQL Server	0,036	0,053	0,381	3,741	36,672	376,749
MySQL	0,034	0,052	0,390	4,224	41,030	405,539

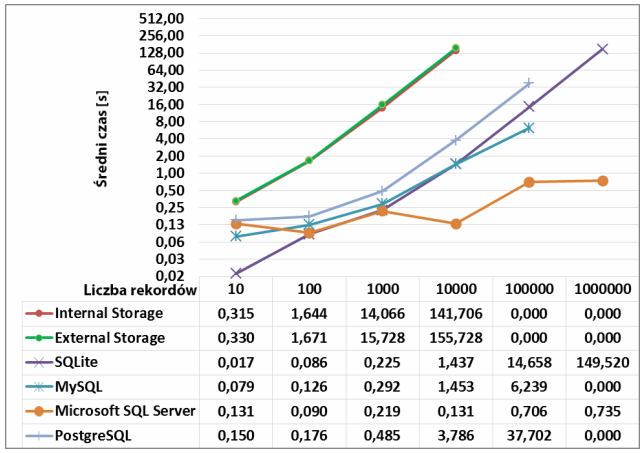
Rys. 6. Wyniki badań czasów odczytu rekordów z danymi tekstowymi na systemie Windows Mobile

Rysunek 7 i rysunek 8 przedstawia wyniki tego badania na systemie Android.



Liczba rekordów	10	100	1000	10000	100000	1000000
Internal Storage	0,020	0,074	0,224	1,055	9,588	102,443
External Storage	0,101	0,146	0,318	1,833	17,862	186,895
SQLite	0,267	1,023	9,968	89,537	165,692	0,000
MySQL	0,095	0,138	0,356	0,593	10,818	0,000
Microsoft SQL Server	0,055	0,072	0,115	0,248	10,756	120,293
PostgreSQL	0,170	0,161	0,162	0,416	4,592	88,177

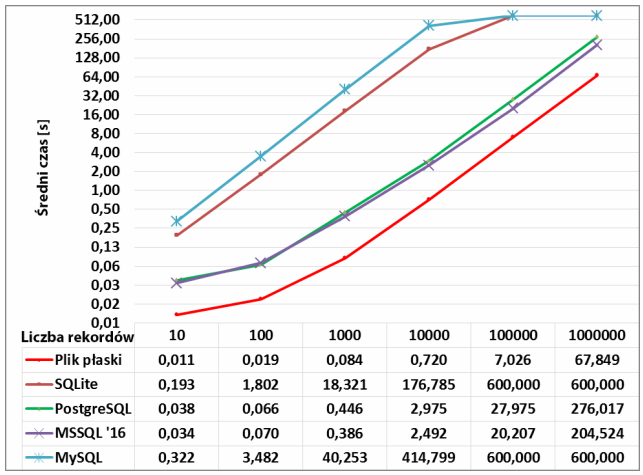
Rys. 7. Wyniki badań czasów zapisu rekordów z danymi tekstowymi na systemie Android



Liczba rekordów	10	100	1000	10000	100000	1000000
Internal Storage	0,315	1,644	14,066	141,706	0,000	0,000
External Storage	0,330	1,671	15,728	155,728	0,000	0,000
SQLite	0,017	0,086	0,225	1,437	14,658	149,520
MySQL	0,079	0,126	0,292	1,453	6,239	0,000
Microsoft SQL Server	0,131	0,090	0,219	0,131	0,706	0,735
PostgreSQL	0,150	0,176	0,485	3,786	37,702	0,000

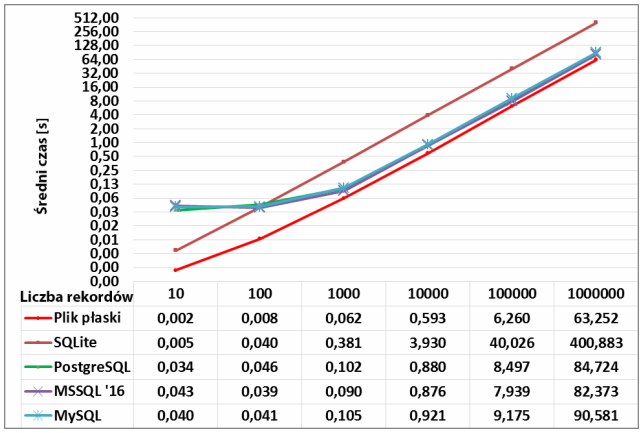
Rys. 8. Wyniki badań czasów odczytu rekordów z danymi tekstowymi na systemie Android

Rysunek 9 oraz rysunek 10 prezentują wyniki zapisu oraz odczytu rekordów z danymi numerycznymi na systemie Windows Mobile.



Liczba rekordów	10	100	1000	10000	100000	1000000
Plik płaski	0,011	0,019	0,084	0,720	7,026	67,849
SQLite	0,193	1,802	18,321	176,785	600,000	600,000
PostgreSQL	0,038	0,066	0,446	2,975	27,975	276,017
MSSQL '16	0,034	0,070	0,386	2,492	20,207	204,524
MySQL	0,322	3,482	40,253	414,799	600,000	600,000

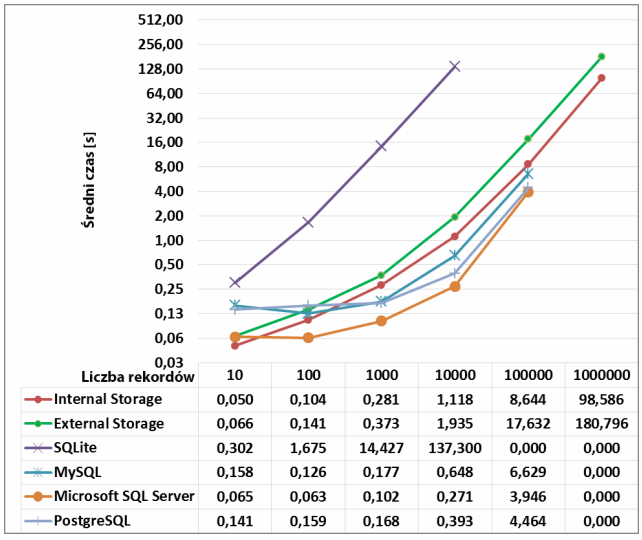
Rys. 9. Wyniki badań czasów zapisu rekordów z danymi numerycznymi na systemie Windows Mobile



Liczba rekordów	10	100	1000	10000	100000	1000000
Plik płaski	0,002	0,008	0,062	0,593	6,260	63,252
SQLite	0,005	0,040	0,381	3,930	40,026	400,883
PostgreSQL	0,034	0,046	0,102	0,880	8,497	84,724
MSSQL '16	0,043	0,039	0,090	0,876	7,939	82,373
MySQL	0,040	0,041	0,105	0,921	9,175	90,581

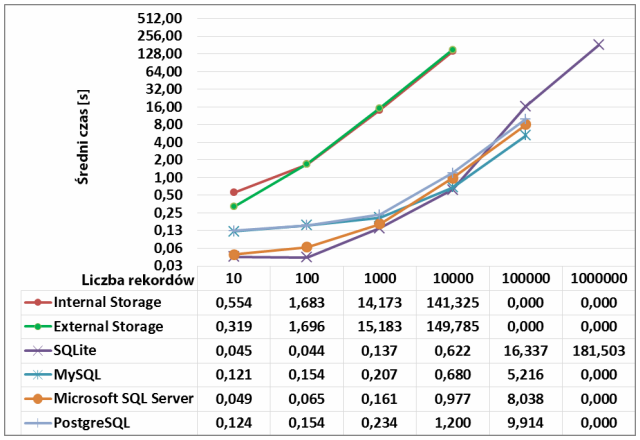
Rys. 10. Wyniki badań czasów odczytu rekordów z danymi numerycznymi na systemie Windows Mobile

Dwa kolejne wykresy widoczne na rysunku 11 i rysunku 12 to wyniki dla kolejno zapisu oraz odczytu rekordów z danymi numerycznymi na systemie Android.



Liczba rekordów	10	100	1000	10000	100000	1000000
Internal Storage	0,050	0,104	0,281	1,118	8,644	98,586
External Storage	0,066	0,141	0,373	1,935	17,632	180,796
SQLite	0,302	1,675	14,427	137,300	0,000	0,000
MySQL	0,158	0,126	0,177	0,648	6,629	0,000
Microsoft SQL Server	0,065	0,063	0,102	0,271	3,946	0,000
PostgreSQL	0,141	0,159	0,168	0,393	4,464	0,000

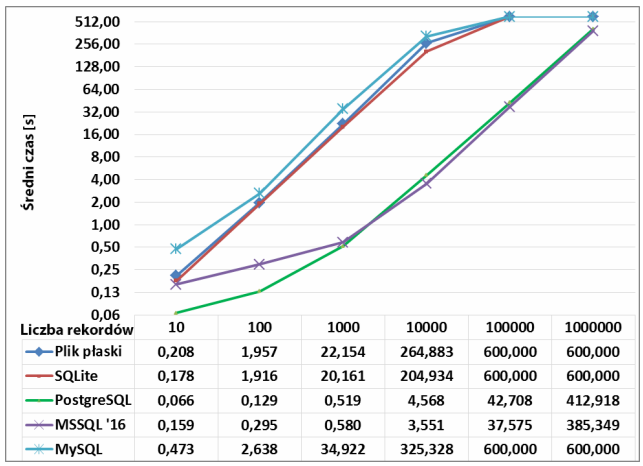
Rys. 11. Wyniki badań czasów zapisu rekordów z danymi numerycznymi na systemie Android



Liczba rekordów	10	100	1000	10000	100000	1000000
Internal Storage	0,554	1,683	14,173	141,325	0,000	0,000
External Storage	0,319	1,696	15,183	149,785	0,000	0,000
SQLite	0,045	0,044	0,137	0,622	16,337	181,503
MySQL	0,121	0,154	0,207	0,680	5,216	0,000
Microsoft SQL Server	0,049	0,065	0,161	0,977	8,038	0,000
PostgreSQL	0,124	0,154	0,234	1,200	9,914	0,000

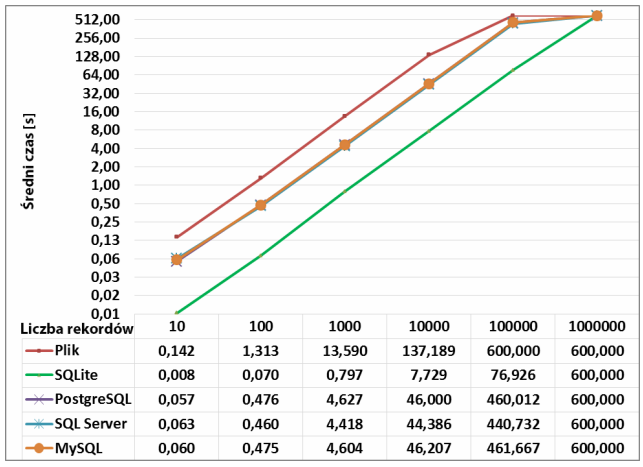
Rys. 12. Wyniki badań czasów odczytu rekordów z danymi numerycznymi na systemie Android

Wykresy z rysunku 13 do rysunku 15 włącznie dotyczą pracy na rekordach zawierających dane binarne. Rysunek 13 oraz rysunek 14 przedstawiają czasy uzyskiwane na systemie Windows Mobile.



Liczba rekordów	10	100	1000	10000	100000	1000000
Plik płaski	0,208	1,957	22,154	264,883	600,000	600,000
SQLite	0,178	1,916	20,161	204,934	600,000	600,000
PostgreSQL	0,066	0,129	0,519	4,568	42,708	412,918
MSSQL '16	0,159	0,295	0,580	3,551	37,575	385,349
MySQL	0,473	2,638	34,922	325,328	600,000	600,000

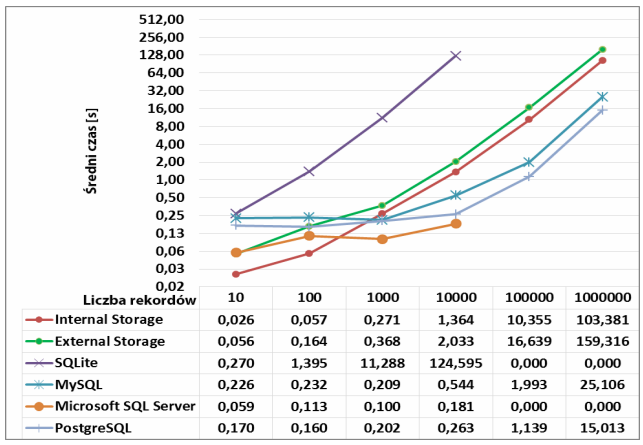
Rys. 13. Wyniki badań czasów zapisu rekordów z danymi binarnymi na systemie Windows Mobile



Liczba rekordów	10	100	1000	10000	100000	1000000
Plik	0,142	1,313	13,590	137,189	600,000	600,000
SQLite	0,008	0,070	0,797	7,729	76,926	600,000
PostgreSQL	0,057	0,476	4,627	46,000	460,012	600,000
SQL Server	0,063	0,460	4,418	44,386	440,732	600,000
MySQL	0,060	0,475	4,604	46,207	461,667	600,000

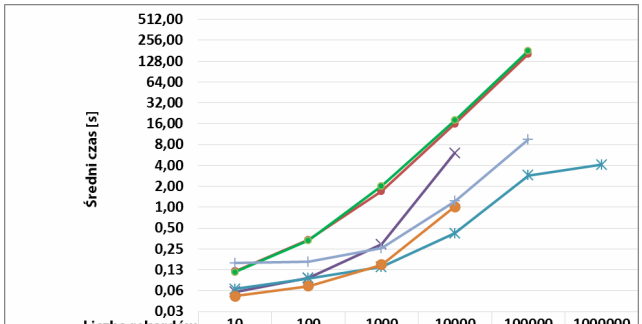
Rys. 14. Wyniki badań czasów odczytu rekordów z danymi binarnymi na systemie Windows Mobile

Rysunek 15 oraz 16 to wyniki tego samego badania na systemie Android.



Liczba rekordów	10	100	1000	10000	100000	1000000
Internal Storage	0,026	0,057	0,271	1,364	10,355	103,381
External Storage	0,056	0,164	0,368	2,033	16,639	159,316
SQLite	0,270	1,395	11,288	124,595	0,000	0,000
MySQL	0,226	0,232	0,209	0,544	1,993	25,106
Microsoft SQL Server	0,059	0,113	0,100	0,181	0,000	0,000
PostgreSQL	0,170	0,160	0,202	0,263	1,139	15,013

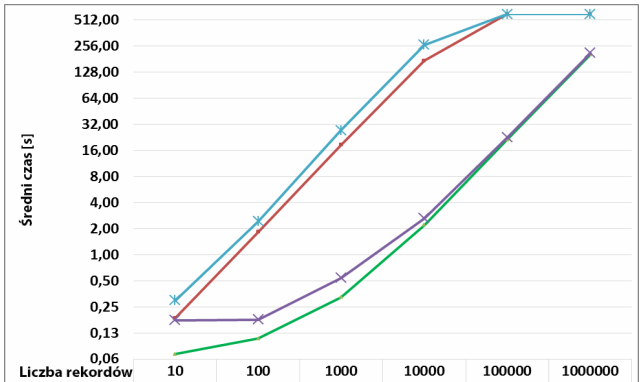
Rys. 15. Wyniki badań czasów zapisu rekordów z danymi binarnymi na systemie Android



Liczba rekordów	10	100	1000	10000	100000	1000000
Internal Storage	0,120	0,339	1,703	15,786	161,564	0,000
External Storage	0,115	0,331	2,018	17,821	180,128	0,000
SQLite	0,060	0,095	0,290	6,041	0,000	0,000
MySQL	0,066	0,094	0,137	0,420	2,872	4,110
Microsoft SQL Server	0,052	0,072	0,148	1,009	0,000	0,000
PostgreSQL	0,156	0,162	0,256	1,218	9,473	0,000

Rys. 16. Wyniki badań czasów odczytu rekordów z danymi binarnymi na systemie Android

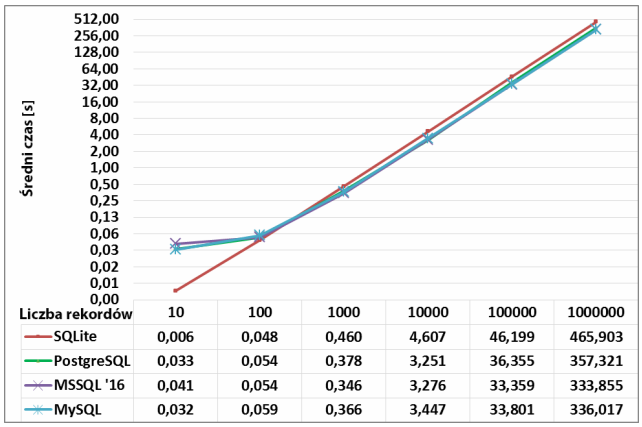
Ostatnią kategorię wyników stanowią wyniki badań czasów zapisu i odczytu rekordów z danymi o różnych typach. Na rysunku 17 widoczne są wyniki na systemie Windows Mobile uzyskane podczas zapisu danych.



Liczba rekordów	10	100	1000	10000	100000	1000000
SQLite	0,185	1,826	18,456	173,163	600,000	600,000
PostgreSQL	0,071	0,108	0,323	2,191	21,284	205,225
MSSQL '16	0,175	0,178	0,540	2,637	22,552	212,789
MySQL	0,297	2,417	27,305	265,329	600,000	600,000

Rys. 17. Wyniki badań czasów zapisu rekordów z danymi o różnych typach na systemie Windows Mobile

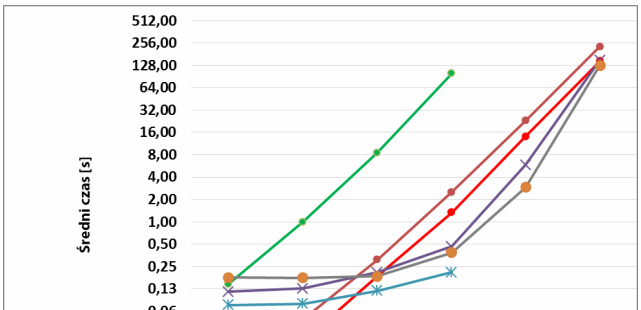
Rysunek 18 przedstawia wyniki odczytu tych danych.



Liczba rekordów	10	100	1000	10000	100000	1000000
SQLite	0,006	0,048	0,460	4,607	46,199	465,903
PostgreSQL	0,033	0,054	0,378	3,251	36,355	357,321
MSSQL '16	0,041	0,054	0,346	3,276	33,359	333,855
MySQL	0,032	0,059	0,366	3,447	33,801	336,017

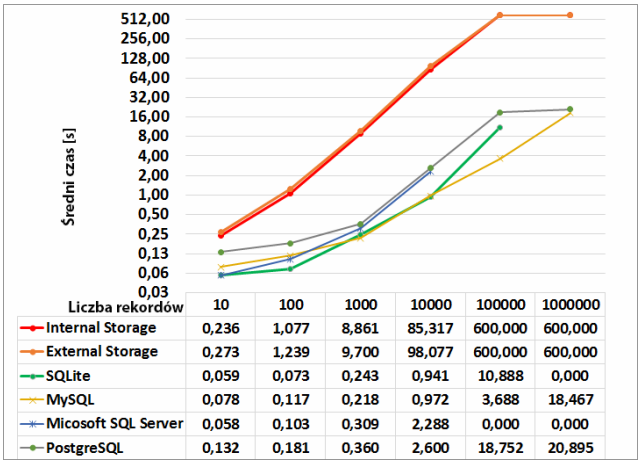
Rys. 18. Wyniki badań czasów odczytu rekordów z danymi o różnych typach na systemie Windows Mobile

Rysunek 19 i rysunek 20 to wyniki zapisu i odczytu rekordów z danymi o różnych typach na systemie Android.



Liczba rekordów	10	100	1000	10000	100000	1000000
Internal Storage	0,019	0,024	0,185	1,329	14,139	146,243
External Storage	0,022	0,048	0,307	2,515	23,021	228,600
SQLite	0,146	0,999	8,528	99,633	0,000	0,000
MySQL	0,114	0,127	0,209	0,467	5,831	150,341
Micosoft SQL Server	0,076	0,079	0,117	0,209	0,000	0,000
PostgreSQL	0,177	0,176	0,186	0,385	2,907	127,819

Rys. 19. Wyniki badań czasów zapisu rekordów z danymi o różnych typach na systemie Android



Liczba rekordów	10	100	1000	10000	100000	1000000
Internal Storage	0,236	1,077	8,861	85,317	600,000	600,000
External Storage	0,273	1,239	9,700	98,077	600,000	600,000
SQLite	0,059	0,073	0,243	0,941	10,888	0,000
MySQL	0,078	0,117	0,218	0,972	3,688	18,467
Micosoft SQL Server	0,058	0,103	0,309	2,288	0,000	0,000
PostgreSQL	0,132	0,181	0,360	2,600	18,752	20,895

Rys. 20. Wyniki badań czasów odczytu rekordów z danymi o różnych typach na systemie Android

5. Wnioski

Należy podkreślić, że nie można wyznaczyć jednoznacznie najlepszej i uniwersalnej formy przechowywania danych na systemach mobilnych. Jak podczas każdego projektu programistycznego, całość sprowadza się do określenia indywidualnych potrzeb stawianych przez wymagania aplikacji oraz wyboru pod tym względem najbardziej odpowiedniego i optymalnego rozwiązania zapewniającego spełnienie określonych przez programistę celów, a także zapewniającego osobie korzystającej z aplikacji odpowiednio wysokie doświadczenie użytkownika.

Istnieją cechy, które są wspólne dla każdej badanej formy przechowywania danych. Mowa tutaj o liczbie rekordów, która wpływa na zwiększenie czasu operacji dodawania oraz wybierania w przypadku każdej badanej formy tj. Local Storage, Shared Preferences, Internal Storage, External Storage, SQLite oraz zewnętrznych baz danych. Najwolniejszą formą przechowywania danych na obu

systemach jest lokalna baza danych SQLite. Kolejną wspólną cechą jest to, że wszystkie formy przechowywania oraz wybierania danych są efektywne do ok. 10 000 rekordów i czas operacji nie przekracza w takich przypadkach kilkunastu sekund. Powyżej tej ilości występują problemy z alokacją pamięci urządzenia, przerwanie połączenia przez zdalny serwer bądź czas operacji wydłuża się do tego stopnia, że może być nieakceptowalny przez użytkownika urządzenia.

W przypadku par klucz-wartość stwierdzono następujące cechy:

- najszybszą formą przechowywania oraz wybierania lokalnych danych formie klucz-wartość jest Shared Preferences na systemie Android oraz LocalSettings na systemie Windows Mobile. Różnica w szybkości wybierania danych potrafi być kilkukrotnie większa na korzyść Shared Preferences w stosunku do External Storage oraz Internal Storage, podobnie sprawa ma się w przypadku LocalSettings w stosunku do pozostałych form przechowywania danych na systemie Windows Mobile;
- najszybszą formą przechowywania oraz wybierania danych na zewnętrznym serwerze jest Microsoft SQL Server, który uzyskał najlepsze czasy do 100 000 tyś rekordów. Powyżej tej ilości wystąpiły problemy z alokacją pamięci na serwerze. W tym przypadku świetnie sprawdził się MySQL, który pomimo najwolniejszych czasów tych operacji w stosunku do pozostałych zdalnych form przechowywania danych pozwolił na bezproblemowe wykonanie badań;
- najwolniejszą formą wybierania danych jest Internal oraz External Storage, w których w każdym z przypadków przy milionie rekordów operacje trwają powyżej 10 minut.

W przypadku rozpatrywania rekordów danych można wyróżnić następujące właściwości:

- najszybszą formą przechowywania lokalnych danych jest Internal Storage;
- najszybszą formą wybierania lokalnych danych typu VARCHAR, NUMERIC oraz rekordów wielotypowych jest SQLite. Dla typu BLOB jest to Internal Storage;
- najwolniejszą formą wybierania lokalnych danych typu VARCHAR, NUMERIC oraz rekordów wielotypowych w formie rekordów jest zarówno Internal Storage jak i External Storage, gdzie operacja wybierania począwszy od 100 000 rekordów trwa powyżej 10 minut. Dla typu BLOB jest to SQLite, który powoduje wyłączenie aplikacji wyrzucając wyjątek już dla 10 000 rekordów przy wielkości danych 10 kB.

Literatura

[1] Hengming F., Jia Ch., Bin X.; The Interaction Mechanism based on JSON for Android Database Application, Academic Journal, 2013 – JSON.

[2] Lee S.; Creating and Using Databases for Android Applications, International Journal of Database Theory and Application Vol. 5 No. 2, 2012.

[3] H.V. Leong and A. Si, Database Caching Over the Air-Storage, The Computer Journal 40(7) , 1997.

[4] Nurseitow N., Paulson M., Reynolds R., Izurieta C.; Comparison of JSON and XML Data Interchange Formats: A Case Study; Montana State University – Bozeman.

[5] Si A., Leong H. L., The Hung Kong Polytechnic University Query optimization for broadcast database, 1998.

[6] Wei J.; Android Database Programming, Packt Publishing Ltd., 2012.

[7] Klasa Cursor, Android Developers, https://developer.android.com/reference/android/database/Cursor.html, dostęp: październik 2016r.

[8] Klasa FileOutputStream, Oracle Help Center, https://docs.oracle.com/javase/7/docs/api/java/io/FileOutputStream.html, dostęp: wrzesień 2016r.

[9] Klasa FileInputStream, Oracle Help Center, https://docs.oracle.com/javase/7/docs/api/java/io/FileInputStream.html, dostęp: wrzesień 2016r.

[10] Klasa SharedPreferences, Android Developers, https://developer.android.com/reference/android/content/SharedPreferences.html, dostęp: wrzesień 2016.

[11] Klasa SQLiteDatabase, Android Developers, https://developer.android.com/reference/android/database/sqlite/SQLiteDatabase.html, dostęp: październik 2016r.

[12] Klasa SQLiteOpenHelper, Android Developers, https://developer.android.com/reference/android/database/sqlite/SQLiteOpenHelper.html, dostęp: październik 2016r.

[13] Save a File on External Storage, Android Developers, https://developer.android.com/training/basics/data-storage/files.html#WriteExternalStorage, dostęp: wrzesień 2016r.

[14] Save a File on Internal Storage, Android Developers, https://developer.android.com/training/basics/data-storage/files.html#WriteInternalStorage, dostęp: wrzesień 2016.