Mobilni i distribuirani informacioni sistemi

- Mobilne baze podataka -

Katedra za računarstvo
Elektronski fakultet u Nišu

Literatura

- *Mobile Developer's Guide To The Galaxy*, 18th Edition, 2019
- *Android Programming The Big Nerd Ranch Guide* , 3rd Edition, Bill Phillips, Chris Stewart, Brian Hardy and Kristin Marsicano, 2017
- *iOS Programming: The Big Nerd Ranch Guide* (6th Edition), A. Hillegass, C. Keur, Big Nerd Ranch Guides; 2017.

Perzistentni podaci u mobilnim aplikacijama

- DBMS – obezbeđuje smeštanje podataka, mehanizme za pristup i alate za upravljanje šemom podataka, mehanizmima pristupa, kao i samim podacima
- Tipovi perzistentnih podataka u mobilnoj aplikaciji
  - *Flat-file* baze podataka kao skup slogova iste vrste
    - Pogodne za jednostavne mobilne aplikacije koje zahtevaju osnovne mehanizme za perzistenciju podataka sa ograničenim mogućnostima u pogledu složenih tipova podataka, kao i pretraživanja i smeštanja velike količine podataka
    - Ne obezbeđuju optimizaciju i poboljšanje performansi, povezivanje i spojeve podataka u različitim datotekama i normalizaciju, poput one u RDBMS)
    - U cilju poboljšanja performansi moguće je određena polja sloga definisati kao ključ, koji se indeksira i po kome je pretraživanje brže
  - Baze podataka (RDBMS, ORDBMS, OODBMS, XDBMS, NoSQL,...)

Mobilne baze podataka

- Baze podataka su osnovna komponenta *enterprise* aplikacija i aplikacija e-poslovanja
- Sa razvojem aplikacija m-poslovanja, baze podataka postaju neizostavne i na mobilnim aplikacijama (pametnim klijentima) i predstavljaju način da *enterprise* podaci „izađu" iz okvira kompanije
- Perzistentni podaci ili *on-line* pristup preko bežične mreže
  - Nedostupna ili nepouzdana bežična mreža
  - Nedovoljan propusni opseg
  - Podaci su uglavnom statički
  - Trajanje baterije je ograničeno
- Hibridno rešenje
  - Koristiti lokalno skladište podataka za pristup podacima neophodnim u mobilnoj aplikaciji,
  - Putem bežičnog Interneta pristupati podacima koji su dinamički ili zavisni od drugih aplikacija

Osnovne karakteristike mobilnih baza podataka

- *Engine* baze podataka i mehanizmi/tehnike koje se koriste za smeštanje podataka
  - Velika količina podataka, brz pristup podacima, indeksiranje, referencijalni integritet, transakcije (ACID), itd.
- Podrška u razvojnim i administratorskim alatima
  - Treba da obezbedi efikasan razvoj i instaliranje (*deployment*) mobilne aplikacije
- Fleksibilna sinhronizacija
  - Poseduje integrisani sloj za sinhronizaciju
- Mali zahtevi za resursima
- Podrška za različite mobilne uređaje, operativne sisteme i platforme
- Podrška za standarde
  - SQL, ODBC, JDBC, itd.
- Sigurnost
  - Zaštita i kriptovanje podataka koji se prenose tokom sinhronizacije, kao i podataka koji su smešteni na uređaju ukoliko je on izgubljen, ili ukraden

Flat-file baze podataka

- PalmOS
- Windows CE
- Symbian OS
- J2ME

PalmOS flat-file baza podataka

- Sadrži i aplikacije i korisničke podatke
- Dve vrste baza podataka: baza podataka slogova (*record*) za smeštanje aplikacionih podataka i resursa (aplikacionog koda i objekata korisničkog interfejsa)
- Baza podataka slogova sadrži zaglavlje baze podataka (*header*) koji sadrži opis baze podataka (ime, atribute, datume kreiranje/modifikacije), *AppInfo* i *SortInfo* blokove, kao i listu slogova (*Record* list) koji sadrži ulaz za svaki slog (ID i atribute sloga)
- Pristup podacima preko PalmOS API
- Konverzija iz/u PalmOS db format
- Sinhronizacija blokova podataka korišćenjem *HotSync*

Windows CE baza podataka

- Baza podataka u okviru *object store* (u okviru koga je i file system i system registry) - CEDB
- Podržava baze podataka smeštene na mobilnom uređaju, ili na eksternim memorijama (memorijskim karticama)
- Pristup putem odgovarajućeg API koji nije kompatibilan sa Win32 database API
- Sinhronizacija korišćenjem *ActiveSync*
- *Embedded database* (EDB) proširenje CEDB koji uključuje
  - Transakcije
  - Korišćenje šeme za predstavljanje strukture baze podataka
  - Pristup od strane više korisnika
  - Sortiranje, indeksiranje, itd.
  - Poboljšane performanse posebno sa velikim bazama podataka

Symbian OS baza podataka

- Relaciona baza podataka
- Pristup preko podskupa SQL i C++ API
- Podrška za transakcije, ali bez mogućnosti spoja tabela
- *Symbian OS Connect* za sinhronizaciju

J2ME Record Management System

- RMS je skup Java klasa za smeštanje i pristup jednostavnim skupovima podataka organizovanim u slogove
- Svaki *RecordStore* predstavlja kolekciju slogova koji su smešteni u trajnu memoriju i koji sadrže kolekciju bajtova
- *MIDlet suite* – grupa MIDlet-a koji pristupaju privatnim *ResordStore* uz mogućnost pristupa deljenim *RecordStore*
- Svaki *RecordStore* ima jedinstveno ime i svakom slogu dodeljuje jedinstveni ID
- API za operacije *open*, *close*, *insert*, *delete*, *move*, itd. nad slogovima
- Neophodno implementirati sopstvene mehanizme za sinhronizaciju

Android - skladištenje podataka

- Android osnovni pristupi za smeštanje i upravljanje podacima - https://developer.android.com/guide/topics/data
  - *SharedPreferences* (parametri podešavanja)
    - Uglavnom služe za smeštanje UI podešavanja u obliku ključ - vrednost
  - Aplikacione datoteke (*App-specific files - Internal/External storage*)
    - Najlakši, ali ne i najbolji način za trajno čuvanje podataka na fajl sistemu
  - *Shared storage*
    - *Podaci koji mogu biti dostupni i drugim aplikacijama i ostaju sačuvani nakon deinstaliranja aplikacije koja ih je kreirala*
  - *SQLite* baza podataka
    - Najlakši i najupotrebljiviji način za čuvanje struktuiranih podataka; svaka aplikacija može da kreira svoju bazu podataka nad kojom ima potpunu kontrolu.
  - Provajderi sadržaja (*Content Providers*)
    - Omogućavaju pristup centralnim skladištima podataka u cilju deljenja podataka između više aplikacija i pristup ovim podacima predstavljenim u tabelarnom obliku.

Shared preferences

- *SharedPreferences API* obezbeđuje skladištenje i čitanje parova ključ-vrednost sastavljenih od primitivnih tipova podataka (boolean, float, int, long, string)
  - Ovi podaci su perzistentni i nakon završetka aplikacije, i brišu se sa deinstaliranjem aplikacije
- Da bi se dobio *SharedPreferences* objekat koristi se *getSharedPreferences()* ako postoji više datoteka sa preferencama, ili samo *getPreferences()* ako je potreban samo jedna datoteka sa preferencama
- Pozivom *edit()* dobija se *SharedPreferences.Editor*, a zatim metodama poput *putBoolean()*, *putInt()*, *putString()*, … unose vrednosti i na kraju sa *commit()* ili *apply()* menja objekat u memoriji koji se upisuje na storage
- Pozivom *getBoolean()* ili *getString()* čitaju se vrednosti

Shared preferences

```
public class Calc extends Activity {
    public static final String PREFS_NAME = "MyPrefsFile";

    @Override
    protected void onCreate(Bundle state){
       super.onCreate(state);
       . . .

       // Restore preferences
       SharedPreferences settings = getSharedPreferences(PREFS_NAME, 0);
       boolean silent = settings.getBoolean("silentMode", false);
       setSilent(silent);
    }

    @Override
    protected void onStop(){
       super.onStop();

      // We need an Editor object to make preference changes.
      // All objects are from android.context.Context
      SharedPreferences settings = getSharedPreferences(PREFS_NAME, 0);
      SharedPreferences.Editor editor = settings.edit();
      editor.putBoolean("silentMode", mSilentMode);

      // Commit the edits!
      editor.commit();
    }
}
```

App-specific files (File API)

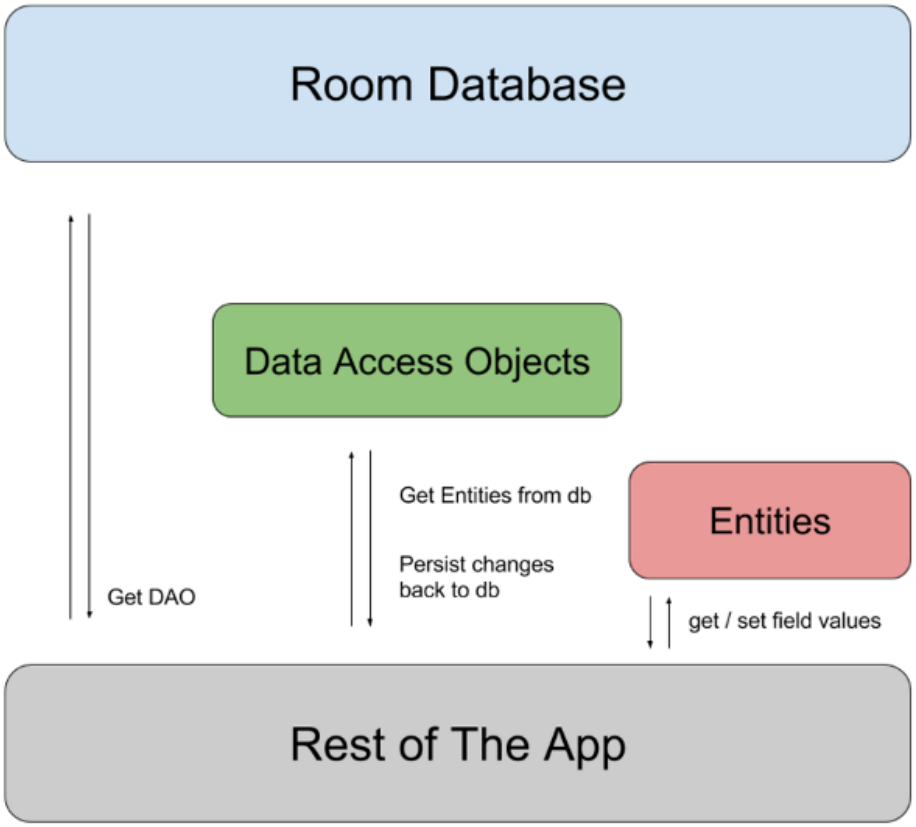
- Interni (*Internal*) storage
  - Uvek je dostupan
  - Datoteke ovde sačuvane dostupne su samo od strane same aplikacije
  - Kada se aplikacija de-instalira sistem uklanja sve datoteke aplikacije
  - Najbolje rešenje kada niti korisnik niti druge aplikacije ne smeju da pristupaju podacima
- Eksterni (*External*) storage
  - Nije uvek dostupan
  - Datoteke ovde sačuvane mogu biti čitane od strane korisnika i drugih aplikacija bez naše kontrole
  - Sistem uklanja datoteke aplikacije samo u određenim slučajevima
  - Najbolje mesto za čuvanje datoteka koje ne zahtevaju restrikcije u pogledu pristupa
    - Datoteke koje želimo da delimo sa drugima aplikacijama
    - Datoteke koje želimo da budu dostupne korisniku
  - Potrebne su *permissions u* manifest-u
  - Pre pristupa eksternom storage-u treba uvek proveriti da li je dostupan (*media mounted*)

SQLite

- Android obezbeđuje punu podršku za SQLite baze podataka
  - Mala baza podataka zasnovana na SQL
  - Koristi standardnu SQL sintaksu
- Baza podataka je dostupna po imenu, u okviru svake klase u aplikaciji, ali ne i izvan aplikacije
- Da bi se kreirala nova baza podataka potrebno je definisati klasu koja nasleđuje *SQLiteOpenHelper* i predefinisati metod *onCreate()* kojim se izvršavaju SQLite komande za kreiranje tabela
- Osnovne CRUD operacije
  - insert(), query(), delete(), update()
- Room – biblioteka koja obezbeđuje sloj apstrakcije preko SQLite (ORM)

Room

- Room obezbeđuje sloj apstrakcije iznad SQLite za jednostavniji pristup SQLite bazi podataka, kao i objektno-relaciono mapiranje.



Content provider

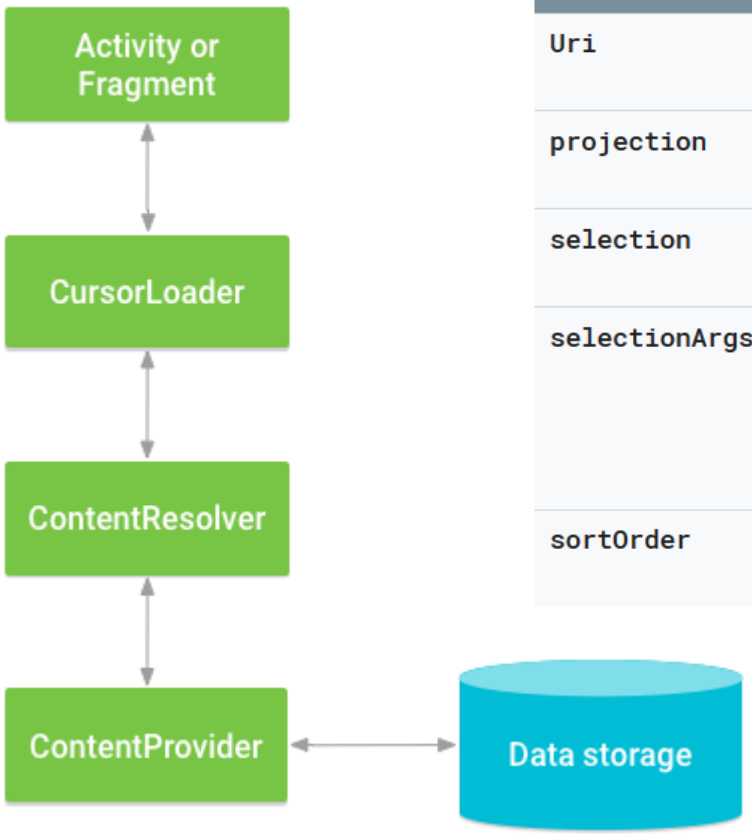
- Omogućavaju pristup centralnim skladištima podataka u cilju deljenja podataka između više aplikacija i pristup ovim podacima predstavljenim u tabelarnom obliku.
- Enkapsuliraju podatke i obezbeđuju mehanizme za definisanje sigurnosti podataka
- *Content provider* obezbeđuje podatke eksternim aplikacijama u vidu jedne ili više tabela slično tabelama u relacionim bazama podataka
- Predstavlja interfejs za međuprocesnu komunikaciju preko zajedničkog repozitorijuma
- Aplikacije moraju da zahtevaju specijalne *permissions* u manifestu za pristup provider-ima
- Android uključuje *Content provider*-e koji upravljaju podacima poput audio, video, slika, i personalnih kontakata

Content provider

- Primarno su namenjeni za korišćenje od strane drugih aplikacija koje pristupaju *provider*-u korišćenjem *provider client* objekta
- *Provider*-i i *provider client*-i obezbeđuju konzistentan, standardni interfejs prema podacima kojim je omogućena i inter-procesna komunikacija i siguran pristup podacima
- Aplikacija pristupa podacima *Content provider*-a korišćenjem *ContentResolver client* objekta
- Obezbeđuje osnovne "CRUD" operacije (*create*, *retrieve*, *update*, *delete*)
- *ContentResolver* uključuje metode u okviru kojih se pozivaju identično nazvani metodi *ContentProvider*-a
- *ContentProvider* predstavlja sloj apstrakcije iznad repozitorijuma sa podacima

Content provider

- mCursor= getContentResolver().query



query() argument	SELECT keyword/parameter	Notes
`Uri`	`FROM table_name`	`Uri` maps to the table in the provider named *table_name*.
`projection`	`col,col,col,...`	`projection` is an array of columns that should be included for each row retrieved.
`selection`	`WHERE col = value`	`selection` specifies the criteria for selecting rows.
`selectionArgs`	(No exact equivalent. Selection arguments replace ? placeholders in the selection clause.)	
`sortOrder`	`ORDER BY col,col,...`	`sortOrder` specifies the order in which rows appear in the returned `Cursor`.

Firebase - A comprehensive mobile development platform

Build better apps

Cloud Firestore
Store and sync app data at global scale

ML Kit BETA
Machine learning for mobile developers

Cloud Functions
Run mobile backend code without managing servers

Authentication
Authenticate users simply and securely

Hosting
Deliver web app assets with speed and security

Cloud Storage
Store and serve files at Google scale

Realtime Database
Store and sync app data in milliseconds

Improve app quality

Crashlytics
Prioritize and fix issues with powerful, realtime crash reporting

Performance Monitoring
Gain insight into your app's performance

Test Lab
Test your app on devices hosted by Google

Grow your business

In-App Messaging
Engage active app users with contextual messages

Google Analytics
Get free and unlimited app analytics

Predictions
Smart user segmentation based on predicted behavior

A/B Testing BETA
Optimize your app experience through experimentation

Cloud Messaging
Send targeted messages and notifications

Remote Config
Modify your app without deploying a new version

Dynamic Links
Drive growth by using deep links with attribution

App Indexing
Drive search traffic to your mobile app

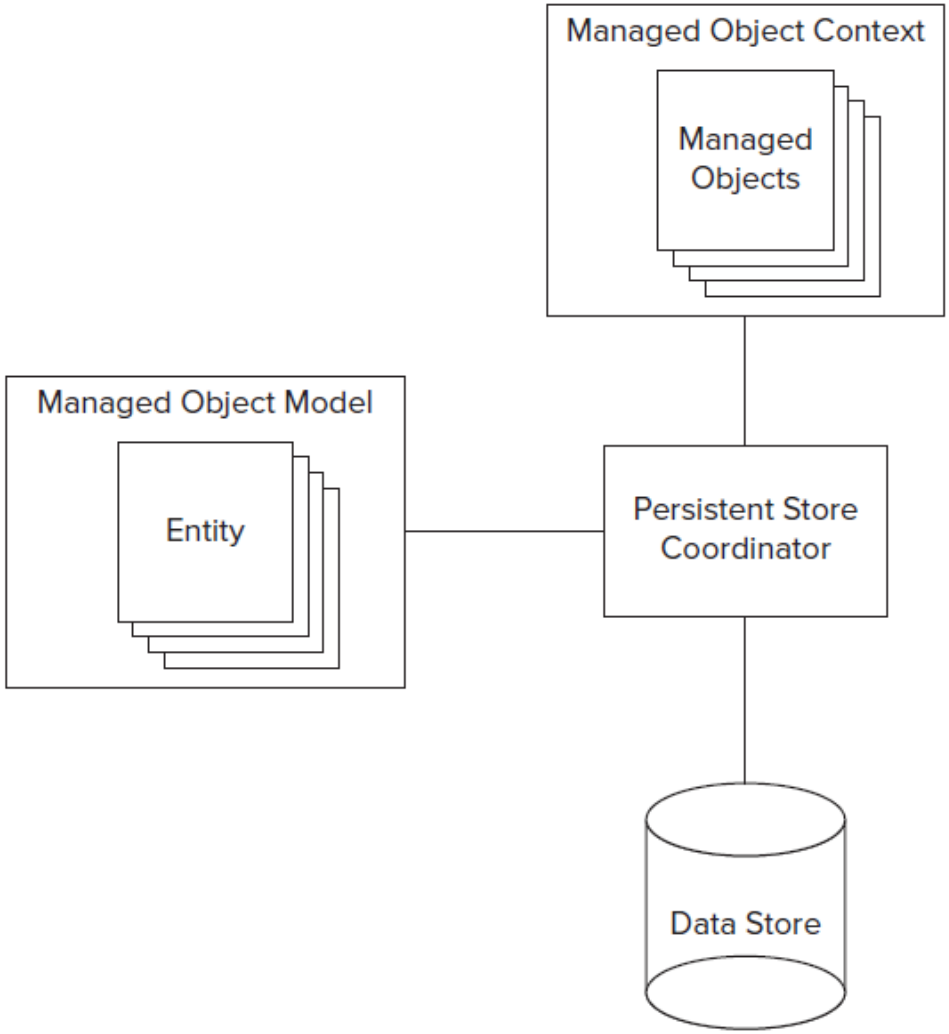
iOS - skladištenje podataka

- iOS upravljanje podacima (*Data management*)
  - https://developer.apple.com/ios/
- Osnovni pristupi za smeštanje i upravljanje podacima
  - *Property List*
    - Uglavnom se koristi za čuvanje specifičnih podešavanja aplikacije u obliku imenovanih vrednosti ili listi vrednosti
  - Smeštanje u *datoteke/direktorijume* (*NSFileManager*, *NSFileHandle* *NSData*) i arhiviranje
  - *SQLite* baza podataka
    - Koristi se za skladištenje podataka u relacionoj bazi podataka i pristup korišćenjem proceduralnog SQL API, poput Android-a
  - *Core Data*
    - Objektno-orijentisani *framework* koji omogućava definisanje objektnog modela podataka aplikacije i upravljanje životnim ciklusom objekata
  - *iCloud* - iCloud Document Storage, iCloud Key-Value Data Storage, CloudKit Data Storage https://developer.apple.com/icloud/

Core Data

- Posebno dizajniran API za smeštanje objekata u Model aplikacije
- *Core Data* i *iCloud* funkcionišu zajedno radi obezbeđivanja infrastrukture za sinhronizaciju podataka na uređajima
- U literaturi često se uz *Core Data* može naići na termine *object persistence framework* ili *object graph manager*
- *Core Data* pruža mogućnost čuvanja modela objekata i mogućnost ponovnog pristupanja kada je to potrebno

Core Data arhitektura



SQLite i Core Data

- *Core Data* može da koristi *SQLite* kao *backing store* (alternativno skladište) za čuvanje podataka
- Pošto *Core Data* može da koristi SQLite, česta greška prilikom razumevanja ovog koncepta je ta da je i sam po sebi relaciona baza podataka – što nije tačno!
- Iako je *Core Data* mnogo prisutniji medju iOS developerima, *SQLite* još uvek ostaje izuzetno korišćen
  - Ukoliko je potrebna funkcionalnost koju pružaju relacione baze podataka i ukoliko se javi potreba za pre-load-ovanjem velike količine podataka na uređaj (primer: GPS navigacija) – rešenje *SQLite*
  - Ukoliko je potrebno da se sačuvaju objekti koje kreira aplikacija tokom izvršavanja - rešenje *Core Data*

Core Data i iCloud

- *iCloud* predstavlja servis kreiran od strane Apple-a koji omogućava korisnicima da automatski sinhronizuju podatke između različitih uređaja
- Omogućava tri tipa skladištenja podataka: *key-value* skladište, skladište bazirano na dokumentima i *CloudKit data* skladište (iOS8)
- Svaka *iCloud-enabled* instanca aplikacije radi sa svojim lokalnim skladištem podataka
- *Core Data* nikad ne sinhronizuje čitavu bazu podataka sa *iCloud-om*.
  - Svaku izmenu *Core Data* zapamti u datoteku, a nakon toga koristi tu datoteku sa zapamćenom promenom kako bi propagirao promene do određenog objekta u lokalnom skladištu

Windows Phone/Mobile - skladištenje podataka

- *Local Folder* (*IsolatedStorage*) – lokalni folder aplikacije
  - Omogućava pogodan način za skladištenje korisničkih podataka u obliku key-value parova unutar *isolated storage* datoteke
  - *IsolatedStorageSettings* - obezbeđuje smeštanje ključ-vrednost parova u *isolated storage*-u.
  - *IsolatedStorageFile* & *IsolatedStorageFileStream* – omogućavaju pristup *isolated storage* prostoru koji sadrži datoteke i direktorijume.
    - Windows Phone 8 koristi *StorageFolder* i *StreamReader* klase
- Lokalna baza podataka za Windows Phone
  - Smeštanje podataka u lokalnu relacionu bazu podataka i pristup korišćenjem LINQ to SQL

DBMS za mobilne uređaje

- SAP (Sybase) SQL Anywhere
  - https://www.sap.com/westbalkans/products/sql-anywhere.html
- IBM Mobile Database (ranije DB2 Everyplace) & IBM solidDB
  - https://www.ibm.com/mobile
- Oracle Database Lite, Berkeley DB
  Oracle Database Mobile Server
  - http://www.oracle.com/technetwork/products/berkeleydb/overview/index.html
  - http://www.oracle.com/technetwork/database/database-technologies/database-mobile-server/overview/index.html
- SQL Server Express (LocalDB) – ranije SQL Server Compact, SQL Server Mobile
  - https://docs.microsoft.com/en-us/sql/database-engine/configure-windows/sql-server-2016-express-localdb
- SQLite - http://www.sqlite.org/

Realm Mobile Database

- Realm Mobile Platform - Realm Mobile Database + Realm Object Server
- Realm Mobile Database – *open source* mobilna baza podataka kao alternativa SQLite i Core Data.
  - https://realm.io/
  - https://realm.io/products/realm-mobile-database/
- GitHub repos - open source: https://github.com/realm
  - realm-core, realm-cocoa, realm-java, realm-js, realm-dotnet,…
- Realm Mobile Platform – obezbeđuje jednostavnu i automatsku sinhronizaciju podataka i upravljanje događajima između Realm Object Server servera i mobilnih uređaja sa Realm Mobile Database
  - https://realm.io/products/realm-mobile-platform/
  - https://realm.io/docs/, https://realm.io/docs/realm-object-server/

Mobilne *enterprise* strategije

- Mobilne strategije u preduzeću
  - *Bring Your Own Device* (BYOD) - *Mobile Application Management (MAM)*
  - *Choose Your Own Device* (CYOD) - *Mobile Device Management & Security* (MDM)
- Neophodne sigurnosne mere
  - Kriptovanje podataka (na uređaju i OTA)
  - Nadgledanje uređaja i daljinsko zaključavanje
  - Zaključavanje/brisanje podataka
- *Enterprise Mobility Management* (EMM) - MAM i MDM provajderi
  - Airwatch - air-watch.com
  - App47 - app47.com
  - Apperian - apperian.com
  - Good - good.com
  - Microsoft - microsoft.com/en-us/cloud-platform/microsoft-intune
  - MobileIron - mobileiron.com
  - Mocana - mocana.com

Mobilne *enterprise* platforme

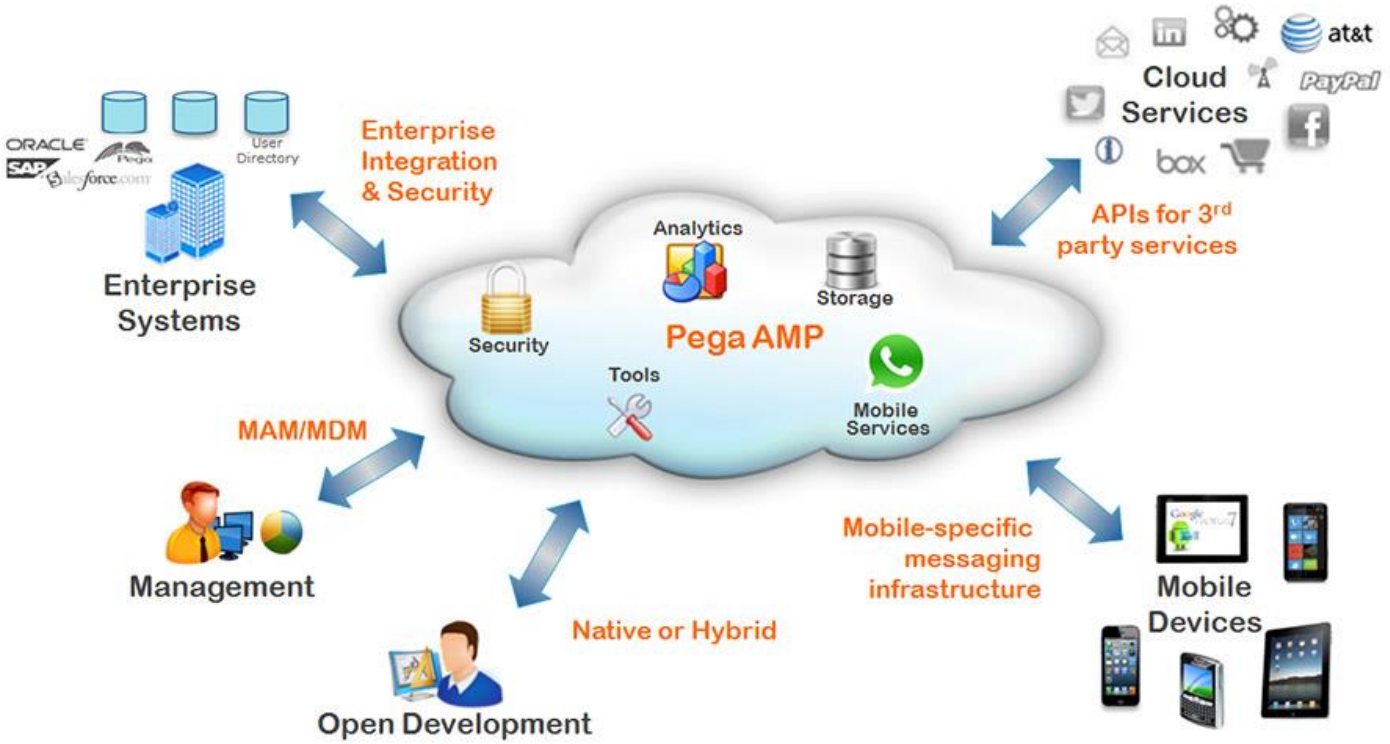
- Ključni elementi mobilnih *enterprise* aplikacija su sinhronizacija podataka i sigurnost
- Mobilna aplikacija mora biti "osvežena" sa relevantnim i ažurnim podacima sa servera kompanije, a takođe i novi ili promenjeni podaci sa mobilnog uređaja treba da budu preneti na server.
- Ograničenje u pristupu podacima je određeno tipom poslovne aktivnosti koju obavlja mobilni službenik, kao i politikom sigurnosti kompanije.
- Sinhronizacija podataka mora biti sigurna, pošto su podaci jedna od najvažnijih svojina kompanije

Mobilne *enterprise* platforme

- Pega AMP (*Application Mobility Platform*)
  - https://www.pega.com/products/pega-platform/mobile
- IBM MobileFirst Platform
  - https://mobilefirstplatform.ibmcloud.com/
- SAP Mobile Platform
  - https://help.hana.ondemand.com/mobile/frameset.htm
- Salesforce Mobile & Salesforce1 for Salesforce App Cloud
  - https://developer.salesforce.com/mobile
  - https://www.salesforce.com/solutions/mobile/app/
- Kony KonyOne
  - www.kony.com/products
- Spring Mobile Solutions
  - www.springmobilesolutions.com

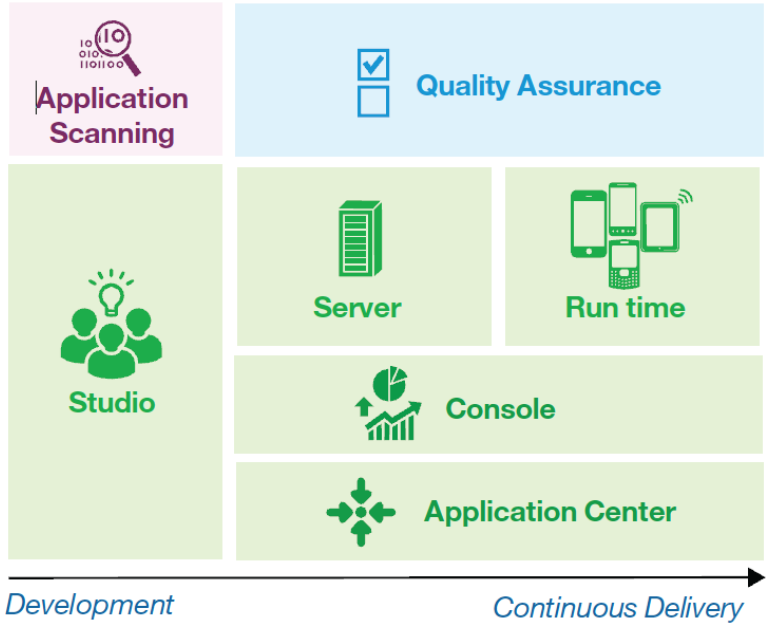
Pega AMP - Services-Based Architecture

- Mobilni servisi:
  - *Enterprise data integration* – mobile messaging (push, pull, request/response)
  - Authentication
  - Notifications
  - Geolocation

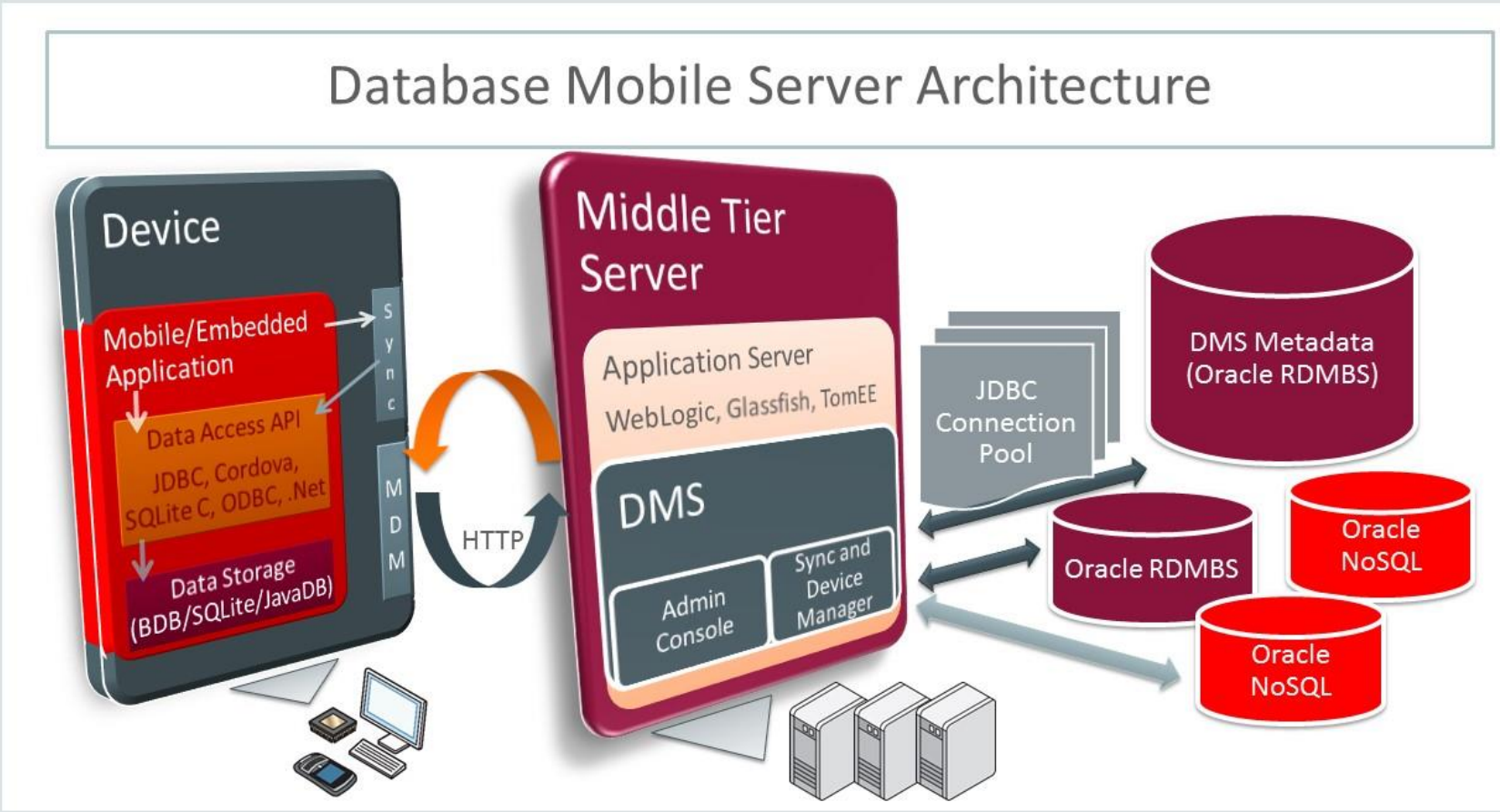


IBM MobileFirst Platform

- IBM MobileFirst Studio proširenje Eclipse IDE za razvoj kros-platformskih mobilnih aplikacija.
- IBM MobileFirst Server - middleware platforma optimizovana za pristup mobilnih aplikacija kao gateway između ovih aplikacija i back-end sistema i servisa.
- IBM MobileFirst Device Runtime Components obezbeđuje runtime API za mobilne aplikacije za proširenje sigurnosti, upravljivosti i upotrebljivosti.
- IBM MobileFirst Application Center omogućava postavljanje *enterprise* app store-a za distribuciju mobilnih *enteprise* aplikacija.
- IBM MobileFirst Console – administratorska GUI konzola za upravljanje serverom, adapterima, aplikacijama i push servisima za pomoć pri upravljanja, nadgledanja i podešavanja mobilnih aplikacija.



Oracle Database Mobile Server



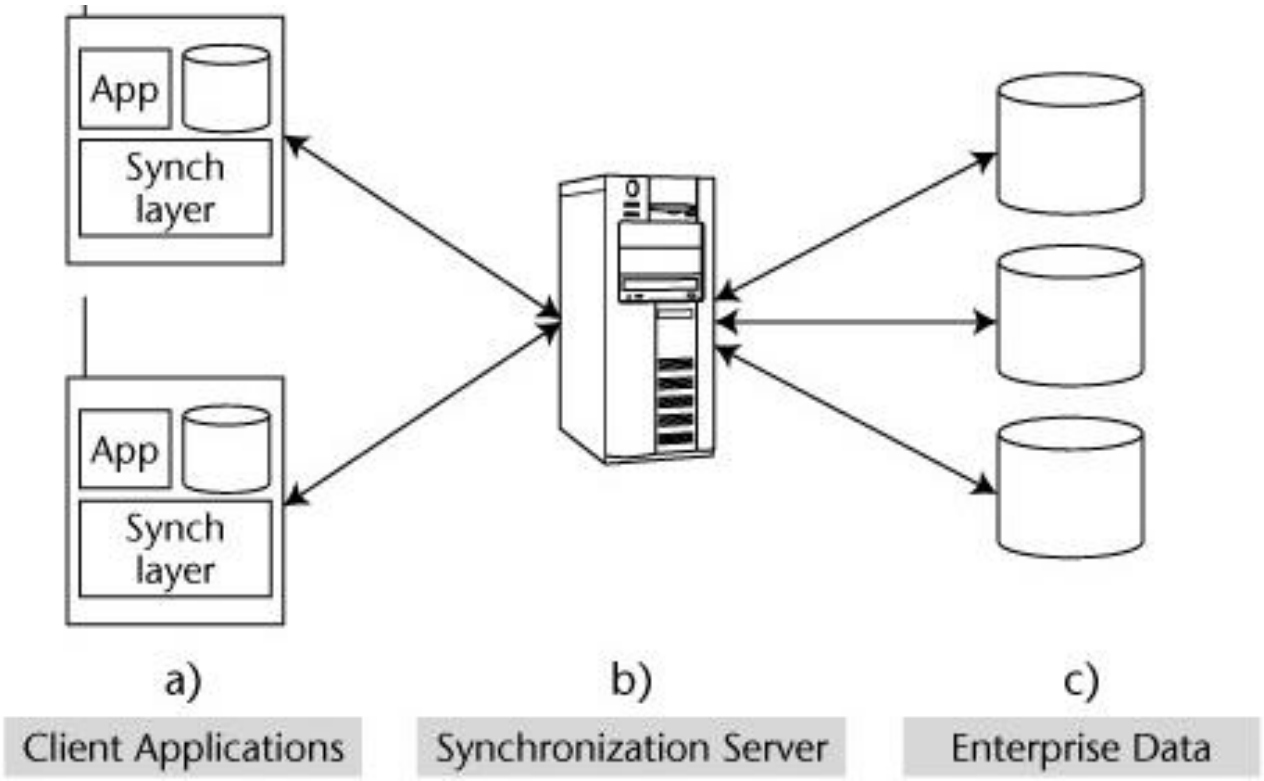
Izvor: Oracle

Sinhronizacija podataka

- Sinhronizacija podataka predstavlja bidirekcionu razmenu i transformaciju podataka između dva odvojena skladišta podataka
- Predstavlja jedan od elemenata integracije *enterprise* aplikacija
- Sinhronizacijom podataka se smanjuje potreba za neprekidnim prenosom podataka preko spore i nepouzdane bežične mreže, tako što se deo podataka smešta lokalno na mobilnom klijentu i periodično sinhronizuje sa podacima na serveru u pogodnim trenucima.
- Takođe omogućuje distribuiranje podataka, a samim tim i aplikacione logike na mobilne uređaje, smanjujući opterećenje servera i veliki broj zahteva od strane mobilnih klijenata.
- Bez sinhronizacije podaci na mobilnom klijentu bi vrlo brzo bili zastareli, a takođe, ukoliko se mobilna aplikacija koristi za prikupljanje podataka sa terena, podaci na serveru bi bili neažurni.

Arhitektura za sinhronizaciju podataka

- Prilikom sinhronizacije svaka promena nad podacima na mobilnom uređaju prenosi se serverskoj bazi podataka, a svaka promena na serverskoj bazi podataka prenosi se mobilnom klijentu i njegovoj lokalnoj bazi podataka. Na taj način su podaci na klijentu i serveru sinhronizovani.

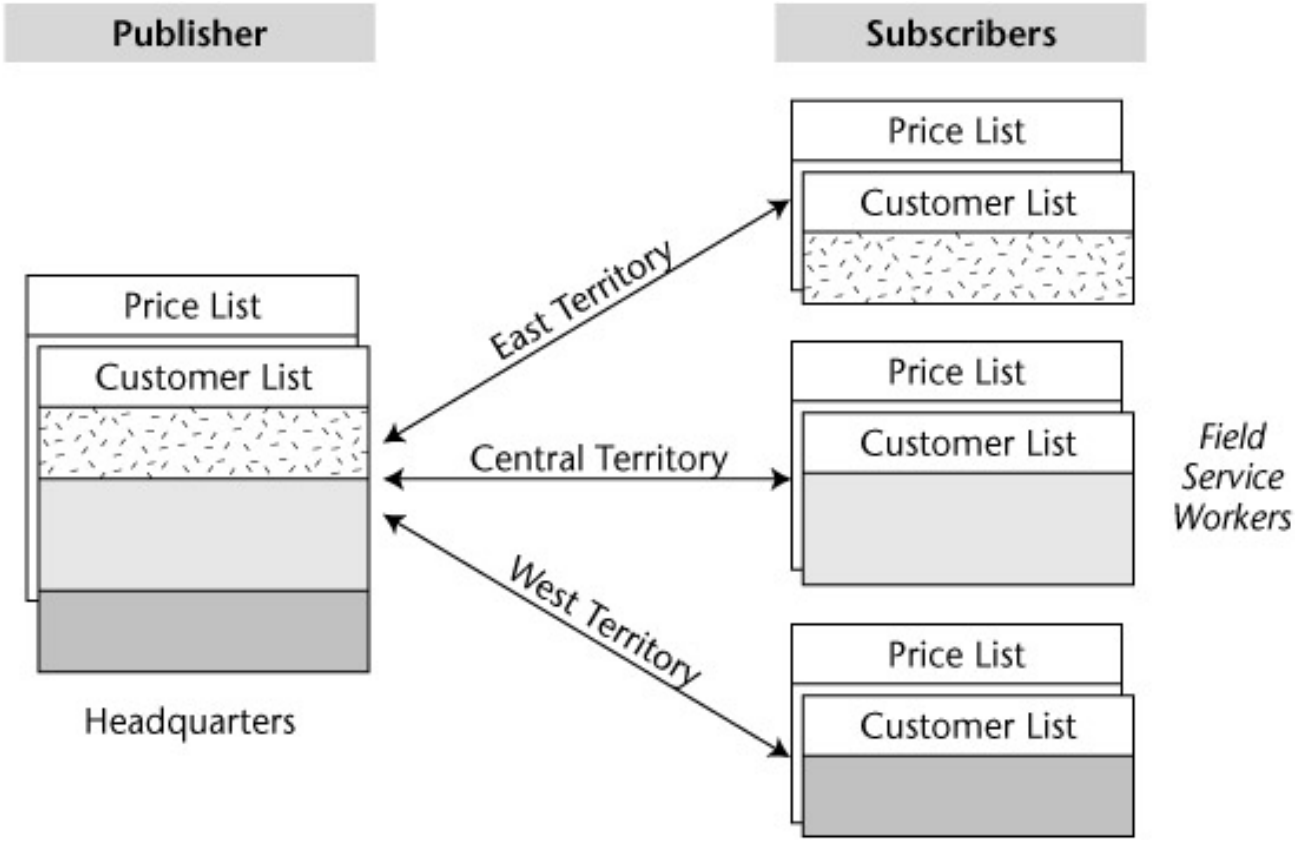


Arhitektura za sinhronizaciju podataka (2)

- Sinhronizacija je najsloženija komponenta mobilne arhitekture sa nativnom ili hibridnom mobilnom aplikacijom ("pametnim" klijentom)
  - Veliki broj mobilnih operativnih sistema, platformi i mobilnih baza podataka, veliki broj različitih bežičnih i žičanih mreža, kao i različite *enterprise* aplikacije i izvori podataka.
- Sinhronizacioni sloj na mobilnom klijentu
  - Komunicira sa sinhronizacionim serverom
  - Treba da ima minimalan uticaj na klijentsku aplikaciju, i da obezbeđuje jednostavan i efikasan API za upravljanje procesom sinhronizacije.
- Sinhronizacioni server predstavlja *middleware* koji sadrži sinhronizacionu logiku
  - Njegove funkcije uključuju: definisanje podskupa podataka, detekcija i rešavanje konflikata u vrednostima podataka, transformacija podataka, kompresija podataka i sigurnost.
  - Funkcioniše u oba smera (bidirekciono)
  - Sadrži i sloj za sinhronizaciju sa *enterprise* aplikacijama i izvorima podataka (RDBMS, CRM, ERP, XML izvori podataka itd.)

Model *Publish/Subscribe*

- Model sinhronizacije podataka Izdavač/Pretplatnik (*publish/subscribe*)

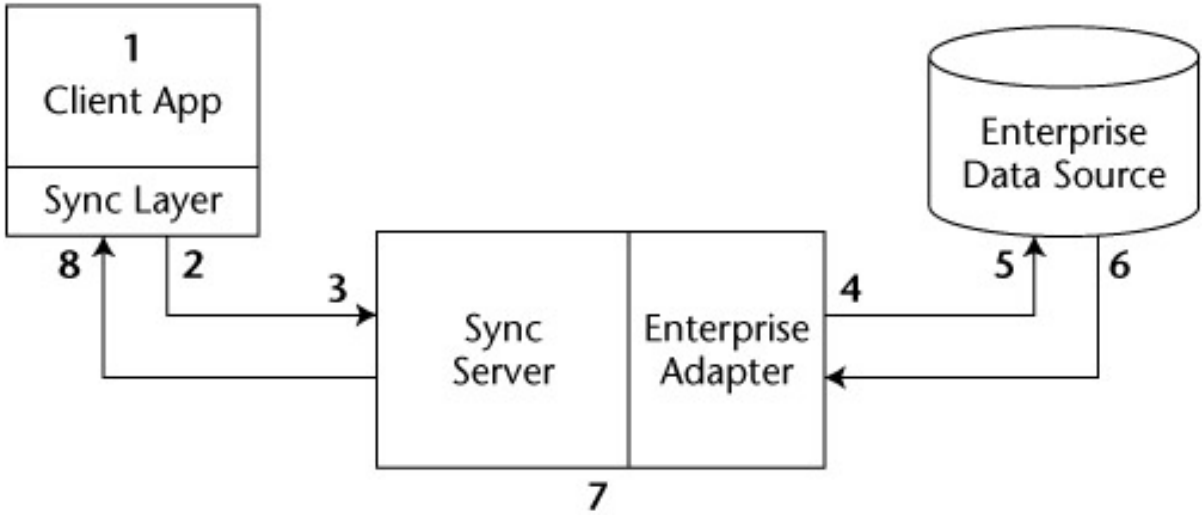


Model Izdavač/Pretplatnik (2)

- Glavnu kopija podataka održava Izdavač (*Publisher*) koji definiše koje tabele, kao i podskupovi tabela su dostupni za izdavanje (publikacije)
- Mobilni klijenti zainteresovani za pristup ovim podacima nazivaju se Pretplatnici (*Subscriber*) koji definišu podatke koje žele da imaju u obliku lokalne kopije
- Različiti podskupovi podataka se publikuju različitim klijentima
- Sinhronizacioni klijent i server održavaju konzistentnim glavnu kopiju podataka i sve njene lokalne kopije

Proces sinhronizacije podataka

1. Sinhronizacioni proces može biti iniciran od strane korisnika aplikacije ili programiran u okviru aplikacije
2. Uspostavlja se konekcija sa sinhronizacionim serverom, autentikuje korisnik i započinje slanje podataka za sinhronizaciju
3. Sinhronizacioni server koristi sinhronizacionu logiku da odredi da li podaci treba da budu transformisani pre slanja do *enterprise* izvora podataka



Proces sinhronizacije podataka (2)

4. *Enterprise* adapter obezbeđuje integraciju sa *enterprise* izvorom podataka (ODBC, JDBC, ADO.NET, itd.)
5. Korišćenjem *enterprise* adaptera sinhronizacioni server vrši autentikaciju korisnika u pristupu *enterprise* izvoru podataka, nakon čega sinhronizacioni server ažurira izvor podataka promenama dobijenim sa klijenta. Ukoliko postoje konflikti u podacima koji se ažuriraju, server preduzima odgovarajuće akcije za rešavanje konflikata
6. Relevantne promene na serveru od poslednje sinhronizacije se pripremaju za slanje klijentu
7. Sinhronizacioni server preuzima promene i obavlja neophodne transformacije, kao i kompresiju i kriptovanje podataka
8. Podatke upućene klijentu preuzima sinhronizacioni klijent koji ažurira lokalnu bazu podataka

Tehnike sinhronizacije

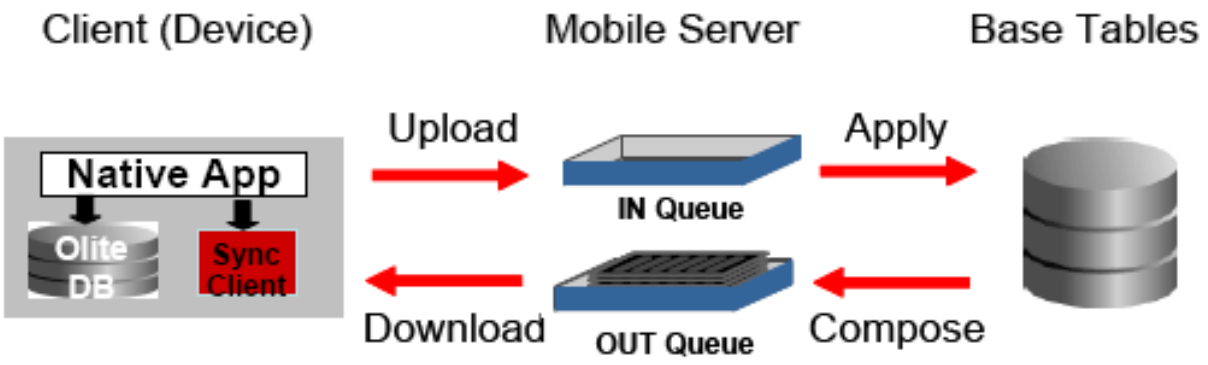
- Modovi sinhronizacije
  - *Snapshot* – slanje nove verzije svih podataka (npr. cela tabela)
  - Samo promene od poslednje sinhronizacije
- Metode za slanje podataka u procesu sinhronizacije
  - Sinhronizacija zasnovana na sesijama
  - Sinhronizacija zasnovana na porukama

Osnovne karakteristike sinhronizacije

- Definisanje podskupa podataka i particionisanje podataka
- Kompresija podataka
- Transformacija podataka
- Integritet transakcija
- Detekcija konflikata
- Rešavanje konflikata
- Podrška za različite mrežne protokole (bežične i žičane)
- Višestruki mehanizmi za prenos podataka
- Integracija sa *enterprise* izvorima podataka i aplikacijama
- Sigurnost

Sinhronizacioni softver

- Sinhronizacioni softverski alati ugrađeni u mobilne operativne sisteme (Windows Mobile ActiveSync, PalmOS HotSync, Symbian OS Connect, itd.)
  - Android Sync Adapters
  - http://developer.android.com/training/sync-adapters/index.html
- Komercijalna rešenja za sinhronizaciju implementiraju kompanije koje razvijaju i mobilne baze podataka (Sybase MobiLink i SQL Remote, IBM DB2 Everyplace Sync Server, Oracle Database Mobile Server, MS SQL Server Mobile/Compact/Express Edition, itd.) i vezana su za mobilne baze podataka te kompanije



Izvor: Oracle

Pitanja i komentari