Forenzička analiza SQLite baze podataka Android uređaja

Soldo, Marko

Master's thesis / Diplomski rad

2020

Degree Grantor / Ustanova koja je dodijelila akademski / stručni stupanj: **University of Zagreb, Faculty of Transport and Traffic Sciences / Sveučilište u Zagrebu, Fakultet prometnih znanosti**

Permanent link / Trajna poveznica: https://urn.nsk.hr/urn:nbn:hr:119:348517

Rights / Prava: In copyright / Zaštićeno autorskim pravom.

Download date / Datum preuzimanja: **2025-03-04**

Repository / Repozitorij:

Faculty of Transport and Traffic Sciences - Institutional Repository

SVEUČILIŠTE U ZAGREBU
FAKULTET PROMETNIH ZNANOSTI

Marko Soldo

FORENZIČKA ANALIZA SQLITE BAZE PODATAKA ANDROID UREĐAJA

DIPLOMSKI RAD

Zagreb, 2020.

SVEUČILIŠTE U ZAGREBU
FAKULTET PROMETNIH ZNANOSTI
POVJERENSTVO ZA DIPLOMSKI ISPIT

Zagreb, 1. travnja 2020.

Zavod: **Zavod za informacijsko komunikacijski promet**
Predmet: **Forenzička analiza informacijsko komunikacijskog sustava**

DIPLOMSKI ZADATAK br. 5896

Pristupnik: **Marko Soldo (0036470849)**
Studij: Promet
Smjer: Informacijsko-komunikacijski promet

Zadatak: **Forenzička analiza SQLite baze podataka Android uređaja**

Opis zadatka:

Objasniti principe i mogućnosti primjene SQLite baze podataka. Analizirati principe forenzičke analize SQLite baze podataka. Napraviti ekstrakciju SQLite baze podataka Android uređaja. Analizirati ekstrahiranu Android SQLite bazu podataka na razini uređaja i aplikacija. Pojasniti oporavak izbrisanih podataka i mogućnosti primjene.

Mentor:

dr. sc. Siniša Husnjak

Predsjednik povjerenstva za diplomski ispit:

SVEUČILIŠTE U ZAGREBU
FAKULTET PROMETNIH ZNANOSTI

DIPLOMSKI RAD

FORENZIČKA ANALIZA SQLITE BAZE PODATAKA ANDROID UREĐAJA

ANDROID SQLITE DATABASE FORENSIC ANALYSIS

Mentor: dr. sc. Siniša Husnjak

Student: Marko Soldo

JMBAG: 0036470849

Zagreb, rujan 2020.

FORENZIČKA ANALIZA SQLITE BAZE PODATAKA ANDROID UREĐAJA

SAŽETAK

Pametni mobilni telefoni danas čine osnovu mnogih ljudskih djelatnosti i aktivnosti. Posljedica toga je kreiranje velike količine podataka. Podaci su često privatne prirode te sadrže osjetljive korisničke informacije. Zbog toga je vrijednost podataka danas veća nego ikad prije. Takav trend pogoduje razvoju digitalne forenzičke analize kao posebne grane forenzičke znanosti. Često su podaci dobiveni metodama forenzičke analize neprocjenjiv izvor dokaza u postupcima kaznenih istraga. U diplomskom radu je obavljen postupak ekstrakcije podataka *Android* pametnog mobilnog telefona. Također, postupak je povezan s internom SQLite bazom podataka u fazi analize. Teorijsku podlogu čini opis interne arhitekture SQLite baze podataka. Osim toga, navedeni su postupci i metodologije procesa forenzičke analize. Cilj diplomskog rada je dobivanje uvida u osnovne mogućnosti *Andriller* i *Autopsy* forenzičkih alata te SQLite baze podataka na primjeru prosječnog korisnika.

KLJUČNE RIJEČI: baza podataka; SQLite; forenzička analiza; Android; Andriller; Autopsy

SUMMARY

Smartphones today make the basis of many human activities. The consequence is the creation of a large amount of data. The data is often private in nature and contains sensitive user information. As a result, the value of data is higher today than ever before. Such a trend favors the development of digital forensic analysis as a special branch of forensic science. Often, data obtained by forensic analysis methods are an invaluable source of evidence in criminal investigation proceedings. In this graduate thesis, the data extraction procedure of an Android smartphone was performed. Also, the process is linked to the internal SQLite database in the analysis phase. The theoretical basis is a description of the internal architecture of the SQLite database. In addition, the procedures and methodologies of the forensic analysis process are listed. The aim of the graduate thesis is to gain insight into the basic capabilities of Andriller and Autopsy forensic tools and SQLite database on the example of the average user.

KEY WORDS: database; SQLite; forensic analysis; Android; Andriller; Autopsy

SADRŽAJ

1. Uvod

Pametni mobilni telefoni su postali dio svakodnevnog života u tolikoj mjeri da ih se smatra glavnim pokretačem nove tehnološke revolucije. Nema sumnje da su ti uređaji promijenili način na koji se obavljaju mnoge aktivnosti i djelatnosti. U nekim slučajevima su postali toliko neophodni da je gotovo nemoguće obaviti potrebne radnje bez njih. To je rezultiralo stvaranjem ogromne količine novih podataka. Često se može čuti izreka koja tvrdi kako će podaci uskoro imati najveću vrijednost u trgovačkom i razvojnom smislu. Rane naznake tog trenda se pokazuju već danas. Naime, postoji mnogo slučajeva gdje su osjetljivi podaci promijenili političke i globalne odnose.

Navedeni trend je zauzvrat doveo do naglog razvoja digitalne forenzičke analize. Radi se o grani forenzičke znanosti koja je usredotočena na izvlačenje i oporavak podataka s digitalnih uređaja za istraživačke, edukacijske ili pravne svrhe. U nekim slučajevima podaci dobiveni metodama forenzičke analize postaju neprocjenjiv izvor dokaza, najčešće u vidu kaznenih istraga. Većina mobilnih telefona je pokretana *Android* operativnim sustavom. Osim toga, *Android* sustav se između ostalog koristi u automobilima, pametnim televizorima, igraćim konzolama i nosivim uređajima. To čini velik skup generiranih podataka što čini odličnu polaznu točku za razvoj forenzičkih metoda i alata.

Upravo zato su poglavlja diplomskog rada usko vezana uz *Android* operativni sustav. Cilj je provesti forenzičku ekstrakciju podataka *Android* uređaja. Osim toga, cilj je povezati analizu s novostečenim znanjem o SQLite bazi podataka. Diplomski rad je podijeljen u sedam cjelina:

1. Uvod
2. Općenito o SQLite bazi podataka
3. Forenzička analiza SQLite baze podataka
4. Ekstrakcija SQLite baze podataka Android uređaja
5. Analiza Android baze podataka na razini uređaja i aplikacija
6. Oporavak izbrisanih podataka i mogućnost primjene
7. Zaključak

Drugo poglavlje je teoretski orijentirano. Radi se o uvodu u baze podataka. Od osnovne terminologije i modela baze podataka do SQLite baze podataka koja posjeduje set specifičnih karakteristika. Osim karakteristika, prikazana je interna struktura SQLite-a što olakšava razumijevanje kasnije provedenih postupaka.

U poglavlju *Forenzička analiza SQLite baze podataka* klasificirane su strukture dnevnika koji se koriste kao pomoćne datoteke prilikom izmjena u bazi podataka. Također njima se postiže integritet koji je isto tako objašnjen u poglavlju. Nakon toga je predstavljen rad sa SQLite alatom i to čini poveznicu teorijskog i praktičnog dijela.

Sljedeće poglavlje daje uvod u metodologiju forenzičke analize mobilnih uređaja i tehnički je orijentirano. Tu je proveden primjer ekstrakcije podataka *Android* uređaja putem odabranog forenzičkog alata.

Peto poglavlje čini poveznicu SQLite baze podataka i forenzičke analize uređaja. Na razini uređaja se kreira izvješće nastalo forenzičkom analizom. S druge strane, na razini aplikacija se rezultati forenzičke analize povezuju s novim znanjem o SQLite bazi podataka čime je omogućen prikaz podataka u tabličnom obliku.

U zadnjem poglavlju prije zaključka su objašnjene mogućnosti oporavka izbrisanih datoteka putem odabranog alata. Također, predstavljena je problematika primjene oporavka u ovisnosti izvora podataka kako bi se dobio dojam o razlici interne i vanjske pohrane *Android* uređaja.

2. Općenito o SQLite bazi podataka

Potreba za pohranom podataka postoji još od razvoja primitivnih trgovačkih sustava. Upravo je sustav trgovine i razmjene dobara bio glavni pokretač razvoja skladišta podataka. Novi izazovi nastaju digitalizacijom i pojavom računalnih sustava koji obrađuju podatke u digitalnom obliku. Mediji za pohranu digitalnih podataka donose određene komplikacije u odnosu na dotadašnje načine pohrane koji su gotovo u potpunosti bili u pisanom obliku pohranjeni u papirnatim arhivama. Osim razumijevanja novih medija za pohranu, nastaje potreba za učenjem novih metoda i tehnologija za pohranu. Sve navedeno je utjecalo na nastanak baza podataka u današnjem obliku.

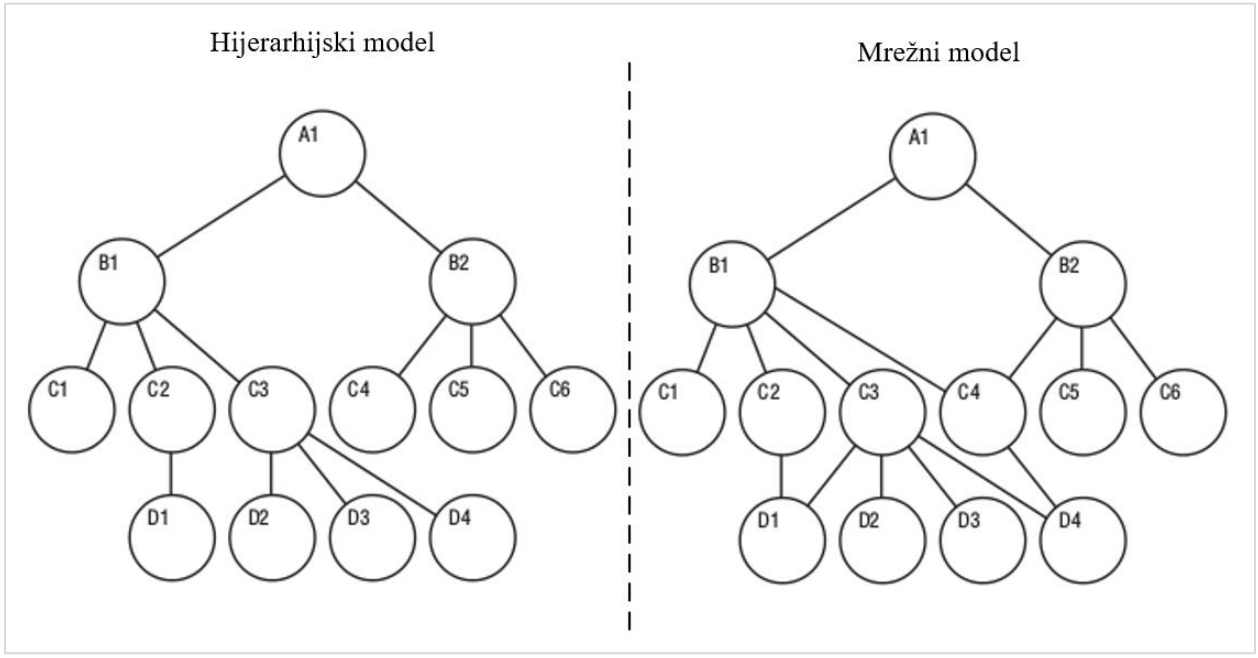
Definicija baze podataka i svih pripadajućih struktura je relativno komplicirana. Ipak, pojednostavljeno se baza podataka definira kao zbirka povezanih datoteka koje sadrže skup podataka i koje su povezane definiranim pravilima. Čak se i povijesni način pohrane analognih podataka može u određenoj mjeri promatrati kao baza podataka. Na primjer, može se zamisliti papirnati arhiv koji sadrži sve kupoprodajne naloge u određenoj trgovini. Isto tako postoje i zapisi u koje su pohranjene informacije o stanju zaliha proizvoda u istoj trgovini. U trenutku obavljanja transakcije potrebno je kreirati novi nalog unutar jednog zapisa te ažurirati stanje zaliha proizvoda unutar drugog zapisa. Naravno, radi se o vrlo jednostavnom primjeru iz kojeg se zaključuje da kompliciraniji sustavi zahtijevaju mnogo veći opseg posla što posljedično vodi do sporijeg obavljanja posla. Upravo zato su današnje baze podataka usko vezane uz upravljački softver koji odrađuje većinu posla prilikom izmjene stanja u bazi podataka. Tako povezana struktura je poznata pod nazivom sustav upravljanja bazama podataka (engl. *DBMS – Database Management System*).

2.1. Razvoj modela baze podataka

Prije pojave današnjih relacijskih baza podataka, jedini način za manipulaciju nad podacima je bio putem nepovezanih datoteka na sličan način kao u ranije navedenom primjeru. Takav način nije bio optimalan za uporabu u kompliciranijim sustavima zato što je prilikom svakog unosa, izmjene ili dohvata podataka bilo potrebno paziti na logičke veze između njih. Razvojem programskih jezika s pravilima za obradu tekstualnih podataka, posao vezan uz pohranu i obradu podataka postaje mnogo lakši za razliku od ranijih ručnih pristupa. Ipak, pristup podacima iz datoteka je i dalje bio kompliciran zadatak. Bez standardiziranog načina pristupa, kakav je danas DBMS, velika je vjerojatnost nastajanja pogreške. Osim toga, takvi sustavi su se vrlo sporo razvijali te je njihovo održavanje bilo vrlo kompliciranо.

Jedan od najvećih problema pri pohrani je redundancija podataka. Naime, bez DBMS-a se podaci često nepotrebno dupliciraju što dovodi do neučinkovitog iskorištavanja ograničenog prostora za pohranu. Redundancija doprinosi i lošoj cjelovitosti podataka tako što se prilikom izmjene često isti podatak ne ažurira na svim mjestima gdje je pohranjen, a to čini bazu nekonzistentnom što krši jedno od osnovnih pravila baze podataka koja su navedena nešto kasnije. Nekonzistentnost baze podataka može dovesti do nesuglasica i što je najvažnije može prouzročiti velike gubitke u poslovanju. Iskustva u takvim slučajevima su bila glavni pokretač razvoja i standardizacije DBMS-a kako bi se osigurao pouzdan način pristupa i ažuriranja podataka. Može se reći kako DBMS čini posrednički sloj između današnjih aplikacija i prostora za pohranu. Na taj način je potrebno poznavati samo rad u SQL-u (*engl. Structured Query Language*) bez detaljnog poznavanja datotečnog pristupa podacima. Tako se smanjuju razlike između stvarnog problema za koji se sustav razvija i tehničke implementacije što smanjuje vremenski opseg posla.

Najraniji model bio je **hijerarhijski model baze podataka** koji se može prikazati strukturom stabla kao na grafičkom prikazu 1. Datoteke i zapisi su povezani u odnosu roditelj-dijete pri čemu se svako dijete može referencirati na samo jednog roditelja. Jasni su razlozi zbog kojih je upravo ovakav model prvi razvijen. To je način na koji funkcionira većina datotečnih sustava. Obično postoji korijen (na grafičkom prikazu označen kao A1) ili direktorij na najvišoj razini koji sadrži više pod-direktorija i krajnjih datoteka. Hijerarhijski model čini veliki napredak u odnosu na rad s nepovezanim datotekama kao što su bili povijesni zapisi na papiru, ali isto tako donosi i neke nedostatke, [1].

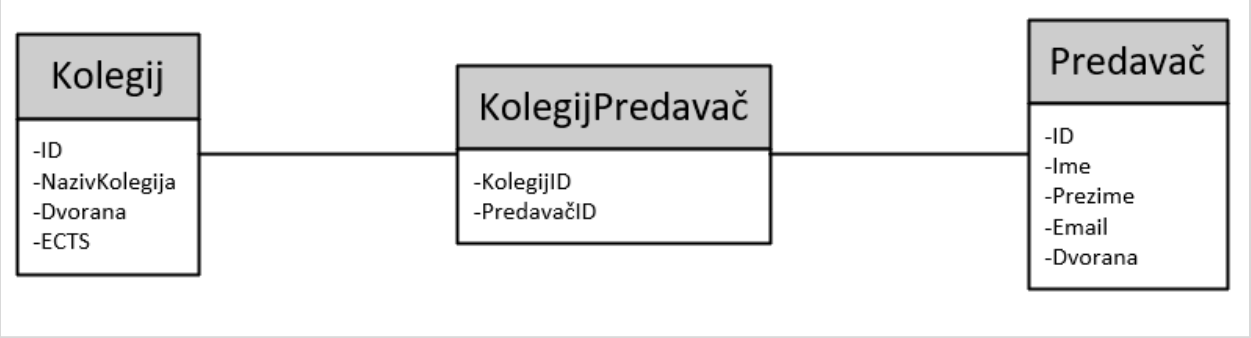


Grafički prikaz 1. Usporedni prikaz hijerarhijskog i mrežnog modela baze podataka
Izvor: [1]

Prvo što je vidljivo iz modela je dobar odnos jedan prema jedan (1:1) i odnos jedan prema više (1:n). Na taj način su određeni sustavi lako ostvarivi. Na primjer, jedan proizvod ima jednu cijenu ili jedan odjel ima više zaposlenih. S druge strane, odnos više prema više (n:n) tu nije ostvariv bez mnogo nepotrebne redundancije. Primjer organizacije fakulteta gdje određeni profesor može predavati više kolegija i gdje određeni kolegij može biti predavan od strane više profesora unutar ovog modela je teško ostvariv. To je bilo jasno iz pravila referenciranja jednog djeteta na isključivo jednog roditelja. Sve navedeno čini ovaj model baze podataka nedovoljno fleksibilnim za većinu današnjih sustava. Isto tako, pristup određenim podacima zahtjeva poznavanje hijerarhije i cijelog puta do podataka. Hijerarhijski model je optimalan za datotečne sustave dok za baze podataka postoje ograničenja.

Mrežni model baze podataka pokazuje određeni napredak u odnosu na hijerarhijski model. Razvijen kako bi riješio nedostatak fleksibilnosti u području odnosa više prema više. Umjesto omogućavanja samo jednog roditelja za svaki čvor, tu se više čvorova može postaviti u ulogu roditelja za pojedini čvor. Na grafičkom prikazu 1 list D4 se referencira na čvorove C3 i C4. Ipak, mrežni model također ima nedostatke. Poboljšanja koje ovaj model donosi nisu opravdana u odnosu na težu implementaciju i održavanje od hijerarhijskog modela. Naime, ovim modelom se još uvijek ne mogu prikazati svi kompleksni odnosi. Težina implementacije se odnosi na programera baze podataka koji mora vrlo dobro razumjeti strukturu modela koja se stalno mijenja pod pretpostavkom da više korisnika koristi sustav, [1].

Većina baza podataka danas su temeljene na **relacijskom modelu.** Naziv je nastao po relacijama kojima su povezani podaci smješteni u strukturama koje su najbolje prikazane u tabličnom obliku. Na grafičkom prikazu 2 je ponuđeno jednostavno rješenje na problem naveden kod opisa hijerarhijskog modela, a odnosi se na implementaciju veze više prema više. Ovaj model je donio veliki napredak u odnosu na mrežni model. Umjesto povezivanja putem odnosa vlasnik-član ili roditelj-dijete, relacijski model omogućuje povezivanje bilo koje datoteke s bilo kojom drugom pomoću zajedničkog polja ili atributa. Tako su na grafičkom prikazu 2 entiteti predavač i kolegij povezani putem atributa ID koji predstavlja jedinstveni identifikator za predavača i kolegij.



Grafički prikaz 2. Primjer relacijskog modela baze podataka
Izvor: Autor (osobni primjer)

Navedeni način povezivanja smanjuje složenost dizajna u velikoj mjeri. Promjene u shemi baze podataka postaju jednostavne i više ne utječu na performanse sustava, ali niti na kompleksnost razumijevanja sheme nakon promjene. Najvažniji čimbenik je način pristupa do datoteka za koji više nije potrebno poznavati kompletan datotečni put nego samo odnose između relacija, a samim time je i mnogo jednostavnije dodavati nove podatke i stvarati odnose između njih.

U početku razvoja relacijski model se činio nepraktičnim. Razlog tome je hardver koji je bio dostupan 1970-ih godina. Naime, relacijski model je pojednostavio uporabu, ali u isto vrijeme pogoršao performanse izvođenja. Razlog tome je bila prilagođenost sustava isključivo hijerarhijskom modelu i datotečnom sustavu, [1]. Od tada je ostvaren ogroman napredak hardvera što je rezultiralo time da danas čak i najjednostavnija računala i terminalni uređaji mogu pokrenuti DBMS. Važna je i poveznica razvoja relacijskog modela i SQL-a. Jednostavnost SQL-a omogućuje čak i početnicima učenje izvođenja osnovnih upita nad bazom podataka u vrlo kratkom vremenskom razdoblju što je velik dio razloga popularnosti relacijskog modela danas.

2.2. Uvod u SQLite

Najjednostavnije rečeno, SQLite je programski (softverski) paket otvorenog koda (*engl. open-source*) koji služi za upravljanje DBMS-om. SQLite je izvorno napisan pomoću programskog jezika C i pripadnih biblioteka pa se često u literaturi definira i kao programski jezik namijenjen za upravljanje podacima unutar DBMS-a, [2]. Za usporedbu, najpoznatiji paketi za upravljanje relacijskim bazama koji su temeljeni na otvorenom kodu su *MySQL* i *PostgreSQL*. Od komercijalne programske podrške ističu se *Oracle Database*, *Microsoft SQL Server* i *IBM DB2* u manjoj mjeri.

Prilikom imenovanja softvera ili općenito proizvoda danas je uobičajen sufiks *Lite* koji obično označava smanjen set mogućnosti u odnosu na matični proizvod. Vrlo važno je shvatiti kako to ovdje nije slučaj. Naime, *Lite* u SQLite označava manju opću složenost, lakšu administraciju i upravljanje te ono što je najvažnije, veoma malu potrebu za resursima uređaja u odnosu na navedenu konkurenciju.

Osim navedenih prednosti, SQLite se najviše razlikuje od ostalih DBMS sustava po komunikacijskoj arhitekturi. Ostali sustavi su kao i većina aplikacija zasnovani na klijentsko-poslužiteljskoj arhitekturi. To uvjetuje izvršavanje svih upita na poslužiteljskom računalu iako je sam program instaliran i na klijentskom računalu. SQLite je pokrenut na uređaju gdje je instaliran i upiti te sva ostala logika obavlja se na klijentskoj strani što ne bi bilo moguće sa zahtjevnijim inačicama DBMS-a (u smislu potrebnih resursa), [3].

SQLite je razvijen 2000. godine. Zamišljen je kao softver potpuno otvorenog koda i zato nije čudno što je danas najraširenija podrška za upravljanje bazom podataka. Razvijen je od strane više volontera, ali najviše zasluga pri razvoju i daljnjem održavanju ima inženjer razvoja softvera dr. Dwayne Richard Hipp, [4].

Kada se govori o raširenosti, pretpostavlja se da je danas u opticaju preko 1 trilijuna (10^{12}) inačica SQLita. Na prvu se čini kao pretjerana procjena, ali kad se u obzir uzme da danas u svijetu postoji 3,5 milijarde pametnih telefona [5] i da svaki uređaj koristi više instanci SQLite-a ovisno o broju instaliranih aplikacija, nije teško doći do toliko velikih brojeva. Osim toga, SQLite je korišten i na mnogo drugih sustava i terminalnih uređaja. Prema [2], SQLite je korišten na većini sljedećih uređaja te interno u sljedećim softverskim rješenjima:

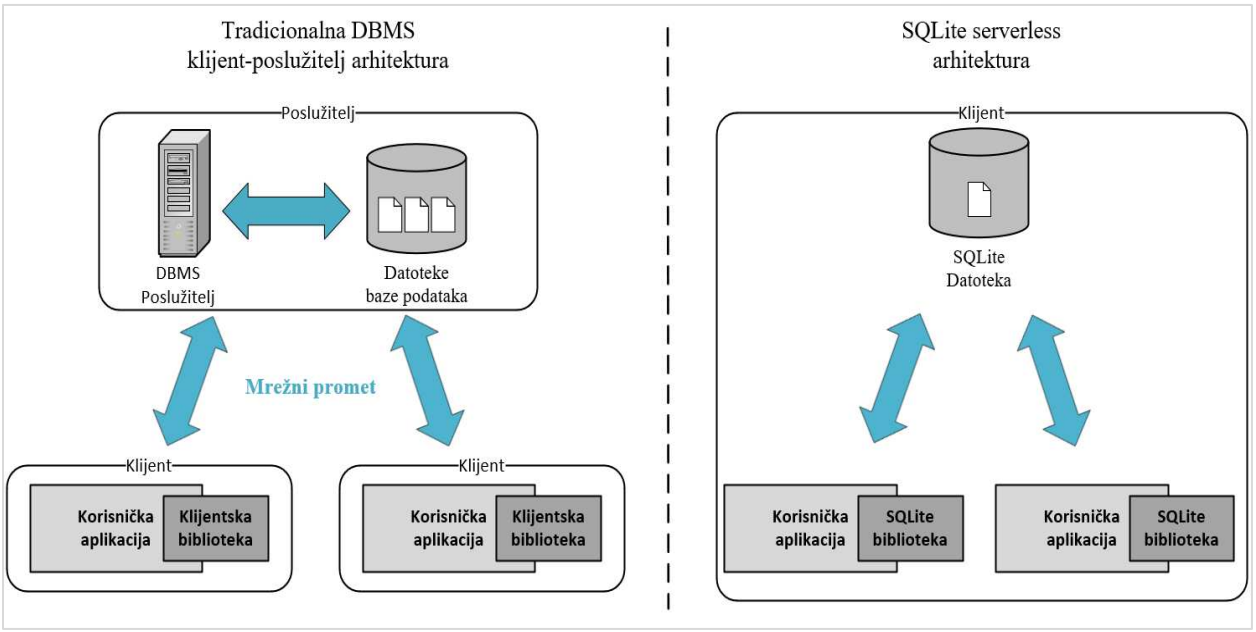
- *Android* i *iOS* uređaji
- *Windows* i *Unix* uređaji
- *Dropbox* klijenti
- STB (*engl. Set Top Box*) uređaji
- Ugradbeni multimedijski sustavi u automobilima
- *Skype*
- *iTunes*
- *Firefox, Chrome* i *Safari* web preglednici

2.3. SQLite karakteristike

Raširenost uporabe SQLite-a je velika zahvaljujući ranije navedenoj komunikacijskoj arhitekturi, ali samo ta karakteristika ne opravdava velike brojeve. U ovom poglavlju su navedene glavne prednosti i karakteristike SQLite-a koje ga čine pogodnim za primjenu na pametnim mobilnim uređajima i ugradbenim sustavima.

SQLite sprema cijelu bazu podataka u jednu datoteku (u literaturi poznato pod nazivom *engl.* ***Single File Database***). U toj datoteci su sadržani izgled i struktura baze podataka, ali i svi podaci. Format datoteke je više-platformski (*engl. Cross-Platform*) što znači da se datoteci može pristupiti na bilo kojem uređaju. Isto tako, prednost spremanja čitave baze u jednu datoteku uvelike olakšava kreiranje, kopiranje i sigurnosno pohranjivanje baze na mediju za pohranu. Kompletna baza podataka se po potrebi može poslati putem mail-a, objaviti na relevantnim web stranicama ili integrirati u sustav za reviziju. Čest je problem oštećenja dijela baze podataka čime cijela baza postaje neuporabljiva, a događa se prilikom krivog ili nepotpunog prijenosa određenih datoteka. Ovdje ne može doći do takvog slučaja i problem manipulacije bazom svodi se na upravljanje jednom datotekom, [6].

Za SQLite **nije potreban poslužitelj (*engl. Serverless*).** Kao što je ranije navedeno, za razliku od većine DBMS-a, SQLite nema klijent-poslužitelj komunikacijsku arhitekturu. Na grafičkom prikazu 3 je vidljiva usporedba takve arhitekture s klasičnom arhitekturom korištenom u većini ostalih DBMS sustava.



Grafički prikaz 3. Usporedni prikaz klijent-poslužitelj i SQLite *serverless* arhitekture
Izvor: [3]

S lijeve strane su klijenti odvojeni od poslužitelja te je potreban mrežni prijenos podataka prilikom operacija nad bazom podataka. Optimizacija i obrada upita obavlja se na strani poslužitelja. Instanca baze podataka sastoji se od velikog broja datoteka pohranjenih unutar datotečnog sustava na poslužitelju. Kako bi aplikacija uspješno pristupila bazi podataka, sve datoteke moraju biti prisutne i ispravne. To komplicira premještanje i sigurnosno pohranjivanje baze podataka. S klijentske strane postoje biblioteke koje sadrže API (*engl. Application Programming Interface*) za pronalaženje i povezivanje s bazom podataka na poslužitelju.

Suprotno tome, s desne strane grafičkog prikaza je vidljivo kako SQLite nema zasebni poslužitelj. Cijela baza je integrirana u jednoj datoteci koja je pohranjena na uređaju i kojoj pristupaju aplikacije što potvrđuje prvu karakteristiku navedenu u ovom poglavlju. Naravno, na uređaju koji koristi SQLite vjerojatno postoji više SQLite datoteka za više aplikacija, ali i više aplikacija može pristupiti jednoj instanci baze podataka. Eliminacijom poslužitelja se kompleksnost pozadinske obrade i provođenja upita znatno smanjuje, [3]. Zato SQLite zahtjeva minimalne funkcionalnosti od uređaja poput sposobnosti čitanja i pisanja u neku vrstu pohrane. Ovakva jednostavnost čini SQLite logičnim odabirom za mnogo okruženja uključujući pametne mobilne telefone, igraće konzole, razne medijske uređaje i ugradbene sustave. S druge strane to znači da SQLite nije dobar u situacijama kada više distribuiranih klijenata treba pristupiti centraliziranoj bazi podataka. U tom slučaju je bolje implementirati jedan od ostalih DBMS-a.

Sljedeća karakteristika pokazuje kako kod SQLite-a nema potrebe za konfiguracijom (*engl.* ***Zero Configuration***). S gledišta krajnjeg korisnika ne zahtijeva se nikakva instalacija niti konfiguracija. Razvojnim inženjerima je dostupan priličan broj parametara, ali oni su skriveni od krajnjih korisnika. Eliminacijom poslužiteljske strane i integracijom DBMS-a u uređaj, korisnici zapravo ne moraju imati osjećaj da koriste bazu podataka.

Jedna od važnijih karakteristika koja je već djelomično spomenuta je **podrška za ugradbene sustave.** Mala veličina koda i minimalna uporaba resursa uređaja pogoduju ugradbenim sustavima (npr. nosivi terminalni uređaji) koji koriste dijelom ograničene operativne sustave. Koristeći zadane postavke, SQLite biblioteka je manja od 700 KB na većini platforma te zahtjeva oko 4 MB memorijskog prostora za rad. Postoji mogućnost izbacivanja nekih naprednijih značajki SQLite-a čime se dodatno smanjuju zahtjevi za pohranu što omogućuje i rad na nekim starijim i ograničenim uređajima (npr. *Symbian* operativni sustav). Nadalje, SQLite kod je pisan na modularan način i zato se očekuje samo minimalna podrška od okruženja sustava. Zato se SQLite danas može pokrenuti na gotovo bilo čemu uz uvjet da ima barem 32-bitni procesor, [7].

SQLite kod **nema korisničku licencu.** Umjesto toga, razvojni tim je odlučio objaviti izvorni kod u javnoj domeni. To znači da su namjerno odustali od bilo kakvih zahtjeva na autorska prava ili vlasništva nad kodom i izvedenim proizvodima. U osnovi, to dopušta svima da rade gotovo bilo što s izvornim kodom, osim naravno da tvrde kako ga posjeduju. Programski kod i stvorene biblioteke mogu se koristiti, modificirati, distribuirati i prodavati na bilo koji način. Također, pazi se da svi komercijalni doprinosi SQLite-u zahtijevaju potpisane izjave u kojima se navodi da autori objavljuju njihov rad u javnu domenu. Sve navedeno osigurava vrlo laku integraciju SQLite-a u sve proizvode uz minimalnu legalnu odgovornost, [8].

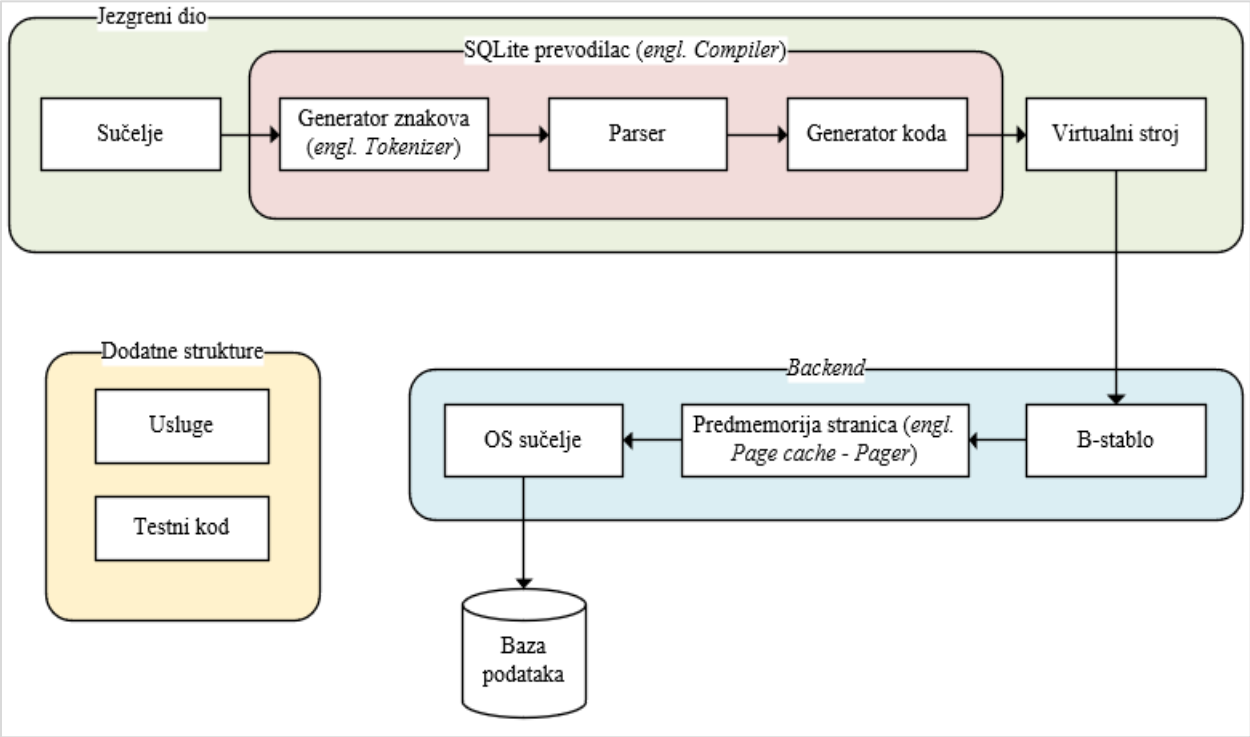
Glavni ciljevi baze podataka su da podaci budu organizirani i sigurni. SQLite pokazuje karakteristike **visoke pouzdanosti.** Za održavanje visoke pouzdanosti provodi se agresivno testiranje jezgrenog koda i osnovnih biblioteka prije objave nove verzije. Standardni SQLite testni paketi sastoje se od preko 10 milijuna testnih jedinica i upita. Testovi uključuju kompletnu pokrivenost svih mogućih upita nad bazom. Također, uključuje testove memorijskog prostora kako se ne bi pojavile kritične greške pri pohrani i dohvatu iz memorije. Testni paket je dizajniran tako da testira izvorni kod do maksimalnih granica što osigurava krajnju pouzdanost pri normalnoj uporabi. Ni jedan softver nije 100% savršen, ali ovakva visoka razina testiranja doprinosi tome da su greške koje pridonose gubitku podataka ili oštećenju baze gotovo nepostojeće, [3].

Skup karakteristika koji ubrzava rad SQLite-a odnosi se na **dinamičnost.** Prema [3], to su sljedeće karakteristike:

- Dinamički način pohrane dozvoljava pohranu bilo koje vrijednosti u bilo koji stupac bez obzira na vrstu podataka
- Mogućnost manipuliranja s više baza podataka istovremeno
- Stvaranje baze podataka u potpunosti u memoriji

2.4. SQLite arhitektura

U prethodnom poglavlju je predstavljena SQLite komunikacijska arhitektura u sklopu predstavljanja jedne od karakteristika. To ne treba miješati s internom SQLite arhitekturom nego služi kao dobar primjer prikazivanja kako ne postoji potreba za poslužiteljem. Na grafičkom prikazu 4 je dana shema interne arhitekture.



Grafički prikaz 4. SQLite interna arhitektura
Izvor: [9]

Ranije je navedeno kako je SQLite izvorni kod pisan na modularan način. Stoga ne čudi modularna arhitektura kojom se ostvaruje jedinstven pristup upravljanju bazom podataka. Osnova arhitekture sastoji se od osam zasebnih modula grupiranih u tri zasebna podsustava kao što je vidljivo na grafičkom prikazu. Takvom modularnom strukturom ostvaruje se podjela upita i instrukcija na diskretne zadatke od kojih se svaki obavlja u zasebnom modulu. Najjednostavnije rečeno, prevodilac sastavlja upit u obliku pogodnom za daljnju obradu, jezgreni dio izvršava upit, a *backend* obrađuje način pohrane i povezuje sučelje s operativnim sustavom, [9].

Ulazni dio u SQLite je definiran jednostavno kao sučelje. Tu se ustvari nalazi API koji čini poveznicu između SQLite-a i programskog jezika C. Taj dio se nalazi u gornjem lijevom kutu grafičkog prikaza i može ga se promatrati kao modul kroz koji sve vanjske biblioteke, programi, programski jezici i skripte ostvaruju komunikaciju sa SQLite-om. Isto tako, to je mjesto putem kojeg programeri, administratori i ostali korisnici komuniciraju sa SQLite-om, [8].

Sljedeća tri modula logički čine jednu cjelinu pa su grupirani u jedan podsustav. Kao što naziv podsustava sugerira, ovdje se naredbe dobivene kroz sučelje prevode u oblik razumljiv slijednim modulima gdje se obavlja daljnja obrada. Postupak prevođenja započinje u generatoru znakova (*engl. Tokenizer*). Taj modul putem sučelja dobiva SQL naredbu ili izjavu u tekstualnom obliku. Potom se vrši validacija sintakse što prevodilac i inače radi u programskim jezicima. Nakon validacije, tekstualni zapis naredbi se dijeli u manje logičke cjeline poznate pod nazivom tokeni. Takve cjeline (tokeni) se prosljeđuju parseru koji pomoću programiranih pravila evaluira tokene i preoblikuje ih u strukturu u obliku liste koja se prosljeđuje u generator koda, [10].

Generator koda razumije strukturiranu listu iz koje generira strojni kod. Taj kod je generiran u takozvanom asemblerskom jeziku koji je zapravo niz instrukcija računalnim registrima i jedinicama, a sljedeći modul (virtualni stroj) upravo kopira ponašanje jednostavnog računala. Asemblerski (*engl. Assembler*) jezik je pohranjen u binarnoj datoteci. Može se reći da je posao generatora koda kreiranje jednog mini-programa pisanog u strojnom (asemblerskom) jeziku te potom predavanje programa virtualnom stroju, [10]. Zaključuje se kako tri navedena modula obavljaju isti posao koji obavlja bilo koji prevodilac programskih jezika.

Izlazni dio jezgre čini virtualni stroj koji je centralni dio arhitekture. U literaturi je poznat pod kraticom VDBE (*engl. Virtual Database Engine*). Virtualni stroj je baziran na registrima što znači da radi s instrukcijama točno definirane dužine. To ne predstavlja problem zato što je asemblerski jezik upravo tako koncipiran da su sve naredbe iste dužine. Time se postiže neovisnost virtualnog stroja u odnosu na operativni sustav i kompletnu arhitekturu uređaja. To je još jedna velika prednost za SQLite pri odluci koji DBMS implementirati u sustav. VDBE je dizajniran posebno i isključivo za obradu podataka te sadrži više od sto mogućih zadataka za rad na bazi podataka. Svaka SQL izjava od odabira i ažuriranja podataka do kreiranja tablica i indeksa najprije se unutar VDBE-a sastavlja u samostalni skup uputa koji definira kako izvesti zadanu naredbu, [10].

Sljedeći podsustav čine struktura binarnog stabla, *pager* i sučelje prema operativnom sustavu. Modul binarnog stabla i *pager* zajedno čine posrednike informacija. Oni prosljeđuju stranice baze podataka, a to su zapravo podatci enkapsulirani u blokove podataka jednake veličine. Unutar stranica se nalaze zapisi podataka i potrebni opisi, [10]. Važno je pravilo kako niti jedan modul nema pristup tim podacima nego samo imaju sposobnost prijenosa tih blokova.

Zadatak modula binarnog stabla je razvrstavanje stranica baze podataka pritom zadržavajući odnose i logičke poveznice između stranica. Stranice su tu organizirane u stablastu strukturu kako bi pretraga bila brža. Ovdje ne treba miješati strukturu stabla s hijerarhijskim modelom. I dalje se radi o relacijskom modelu, samo što je ovdje manji broj stranica baze podataka koje je dobro organizirati u binarno stablo zbog pretrage podataka. Naime, pokazalo se da pretraga podataka po binarnom stablu smanjuje vrijeme potrebno za pretragu u odnosu na slijednu pretragu za više od 70%, [10].

Pager je modul čiji posao je transport i efikasnost. Tu se odvija prijenos stranica baze podataka u oba smjera (prema pohrani i od pohrane prema B-stablu). Modul binarnog stabla povlači stranice iz *pagera*, a s druge strane dobiva skup uputa za operacije nad stranicama iz virtualnog stroja. Operacije s pohranom ili konkretno s diskom za pohranu su danas neke od najsporijih operacija koje uređaj obavlja. Stoga je *pager* dizajniran tako da pokušava ubrzati taj proces zadržavanjem često korištenih stranica u predmemoriji čime se minimalizira potreba za čestim pristupom disku za pohranu, [7].

OS sučelje pruža sloj apstrakcije između operativnog sustava i SQLite modula. Rezultat toga je da ostali moduli vide jedno konzistentno sučelje neovisno o operativnom sustavu uređaja. Prema tome, *pager* ne mora brinuti o nekim operacijama poput zaključavanja datoteka i stranica te da li se te operacije izvode na različit način na *Android* ili *iOS* operativnim sustavima. Za sve to se brine API unutar OS sučelja. Ovaj modul je najzaslužniji za prenosivost SQLite-a između različitih sustava i uređaja, [10].

Razni alati i uobičajene usluge kao što su alokacija memorijskog prostora, usporedba tekstualnih nizova i *Unicode* pretvorba nalaze se u modulu usluga. U osnovi to je modul koji je pozivan za usluge i operacije koje koristi više drugih modula. Modul za testiranje sadrži velik broj testova dizajniranih za ispitivanje svakog malog dijela koda baze podataka. Ovaj modul je glavni razlog zašto je SQLite kod toliko pouzdan. Potpuno regresijsko testiranje se provodi jednostavnim pozivom modula. Važno je znati i da su testovi također otvorenog koda što ih čini dostupnim za uređivanje i prilagođavanje svakoj situaciji, [10].

3. Forenzička analiza SQLite baze podataka

Baza podataka zajedno s DBMS-om čini sloj između prostora za pohranu i aplikacija ili programa na višoj razini. Prilikom redovne uporabe današnjih pametnih mobilnih telefona korisnici nemaju doticaj s bazom podataka i pripadnim strukturama. Čest je slučaj da prilikom postupaka forenzičke analize uređaja također ne postoji potreba za pristupom bazi podataka. Naime, potrebnim podacima se pristupa preko baze podataka, ali većina forenzičkih alata ne zahtijeva detaljno poznavanje materije. Ostvaruje se dojam kako je ekstrakcija podataka ostvarena pomoću datotečnog sustava što je točno u određenoj mjeri, ali ne i sasvim moguće bez DBMS-a kao srednjeg sloja. Prema tome, poznavanje barem osnovnih principa baze podataka s kojom se radi olakšava provođenje forenzičke analize.

3.1. Integritet SQLite baze podataka

Prilikom svih operacija nad skupom podataka važno je osigurati integritet podataka. Pri tome se misli na osiguravanje cjelovitosti, točnosti i dosljednosti podataka. Integritet baze podataka obično se nameće tijekom faze dizajna baze podataka korištenjem standardnih postupaka i pravila. Isto tako, održavanje integriteta tijekom čitavog životnog vijeka baze podataka ostvaruje se korištenjem različitih metoda provjere pogrešaka i provjere valjanosti, [11].

SQLite za očuvanje integriteta osim metoda koristi dodatne strukture i datoteke. Konkretno, radi se o povratnom dnevniku (*engl.* ***Rollback Journal***) i WAL (*engl.* ***Write Ahead Log***) dnevniku. Navedene datoteke pomažu očuvati integritet baze podataka u slučaju nastanka greške prilikom operacije zapisivanja u bazu. U poglavlju SQLite karakteristike je definirano da je SQLite integriran u jednoj datoteci, ali isto tako postoje i radne datoteke koje su privremene, a upravo takve datoteke služe za očuvanje integriteta. Takve datoteke se brišu kada se zatvori aplikacija koja koristi SQLite biblioteke, ali svejedno čine sastavni dio pokrenutih aplikacija, [2]. Forenzički gledano, sadržaj tih datoteka je često od većeg interesa od glavne baze podataka. Razlog tome su slučajevi kada korisnici pokušavaju sakriti tragove aktivnosti brisanjem podataka.

Većina današnjih baza podataka su definirane kao transakcijske baze, a to je slučaj i sa SQLite-om To znači da SQLite grupira skup naredbi u jednu transakciju. Tako grupirane naredbe se potvrđuju i izvršavaju zajedno. Veličina transakcije nije definirana. U poglavlju o SQLite arhitekturi je objašnjeno kako je stranica (ili blok) interna struktura u koju se enkapsuliraju podaci. Zato je najmanja moguća veličina transakcije jednaka veličini stranice čak i u slučaju da se radi o samo jednoj naredbi. Naravno, transakcija može biti veličine više stranica, [2].

Transakcijska baza podataka ima ACID svojstva (*engl. Atomicity, Consistency, Isolation, Durability*). Prema [2], svojstva su sljedeća:

- **Atomarnost** → Sve izmjene u bazi podataka moraju pratiti pravilo *"sve ili ništa"*. Time svaka transakcija postaje jedan atom posla. Ako se pojavi greška u najmanje jednoj naredbi, kompletna transakcija se neće izvršiti. Znači izvršit će se sve naredbe u transakciji ili niti jedna.
- **Konzistencija** → Osigurava se konzistentno stanje baze podataka. U slučaju da se transakcijom pokuša narušiti pravila konzistencije i postavljena ograničenja, DBMS poništava transakciju i vraća bazu u zadnje konzistentno stanje. Kada se transakcija uspješno izvrši DBMS definira novo konzistentno stanje baze podataka.
- **Izolacija** → Bez svojstva izolacije nije moguć višekorisnički rad. Prilikom izvršavanja više transakcija u isto vrijeme osigurava se neometan rad. Znači stanje baze nakon izvršenja više istovremenih transakcija mora biti isto kakvo bi bilo da su se transakcije izvršile slijedno u bilo kojem poretku. Kod SQLite-a se u većini slučajeva radi o jednom korisniku, ali višekorisnički rad je prisutan u slučajevima kada više aplikacija pristupa sistemskim bazama podataka na razini uređaja.
- **Trajnost** → Sve potvrđene transakcije ne smiju biti izgubljene. DBMS osigurava trajnost pomoću struktura. Radi se o zapisu transakcija (*engl. transaction log*) i rezervnim zapisima (*engl. database backup*). Navedene strukture omogućuju obnovu transakcija i baze podataka neovisno o hardverskim i softverskim greškama.

ACID usklađenost baze podataka znači da nakon nepredviđenog događaja (nestanak električne energije, rušenje aplikacije ili operativnog sustava) baza podataka ostaje u stabilnom, važećem i iskoristivom stanju što je preduvjet za integritet.

Vraćanje baze u zadnje konzistentno stanje može značiti gubitak zadnjih nekoliko stranica koje sadržavaju transakcije, ali postiže se očuvanje integriteta što je mnogo važnije. Uostalom, velika je vjerojatnost da će korisnik zapamtiti nekoliko zadnjih promjena prije nastanka greške što mu omogućuje da te promijene ručno ponovi. Sve navedeno se u SQLite-u postiže dnevnicima koji su spomenuti na početku poglavlja, a radi se o privremenim pomoćnim datotekama. Danas je u većini slučajeva WAL dnevnik u standardnoj uporabi, ali povratni dnevnik je prvi razvijen. Ako se eksplicitno ne navede drugačije pri kreiranju nove baze podataka, zadani način rada je povratni dnevnik, [2]. Preporuka je da se uvijek na početku definira način rada kako kasnije ne bi došlo do nepredviđenih situacija. U sljedećim poglavljima je dan osnovni pregled dnevnika.

3.1.1. Povratni dnevnik

Prilikom razvoja SQLite baze podataka za novu aplikaciju prvo se stvara prazna baza. Na početku se naredbama definira interni rad baze i pripadnih biblioteka. Objašnjeno je kako nije potrebno sve definirati eksplicitno, ali je praksa ipak navesti određene definicije. Razlog tome su česte promjene razvojnih inženjera na projektu. Tako se brzim pogledom na definicije mogu dobiti osnovni podaci o radu baze. Korištenje povratnih dnevnika u SQLite-u se može definirati na način prikazan u tekstualnom prilogu 1.

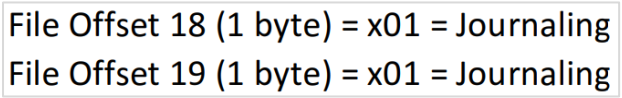
```
1.    PRAGMA journal_mode = DELETE
2.
3.    PRAGMA journal_mode = PERSIST
```

Tekstualni prilog 1. SQLite naredbe – Definicija uporabe povratnog dnevnika
Izvor: [12]

Ključna riječ PRAGMA (od *engl. Pragmatic information*) je namijenjena isključivo SQLite-u i nije dostupna kod ostalih DBMS-a. Koristi se za definiciju i izmjenu zadanog načina rada baze podataka. Pri tome se misli na interni način rada što mijenja samo ne-tablične podatke. PRAGMA naredbe se zadaju putem istog sučelja kao i tipične SQL naredbe (npr. SELECT, INSERT).

U tekstualnom prilogu 1 se nalaze dva najčešće korištena načina za definiciju uporabe povratnih dnevnika. U DELETE načinu rada povratni dnevnik se briše na kraju svake transakcije. Zapravo, operacija brisanja povratnog dnevnika je okidač za potvrđivanje transakcije. PERSIST način rada onemogućuje brisanje dnevnika na kraju transakcije. Umjesto toga, zaglavlje dnevnika se prepisuje nulama. Ovakav način rada je poželjan zbog optimizacije na platformama gdje je brisanje datoteke mnogo sporije od prepisivanja bloka datoteke s nulama. Postoje još dva manje korištena načina rada povratnih dnevnika. Jedan od njih je TRUNCATE koji smanjuje veličinu dnevnika na nulu umjesto brisanja. Drugi je MEMORY koji pohranjuje dnevnik u RAM (*engl. Random Access Memory*) čime se štedi na vremenu potrebnom za pristup disku, [12].

U poglavlju SQLite arhitektura je objašnjen modul B-stabla koji organizira stranice u stablastu strukturu. Korijen stabla čini stranica koja osim pokazivača na stranice sljedeće razine sadrži i zaglavlje. Zaglavlje sadrži zapise pomoću kojih se može razlikovati korišteni dnevnik. Primjer kako izgleda dio zapisa u slučaju korištenja povratnog dnevnika nalazi se na slici 1.



Slika 1. SQLite zaglavlje – Povratni dnevnik
Izvor: [13]

Povratni dnevnik je u SQLite-u implementiran tako da se u njega prvo zapiše kopija stranice koja će biti promijenjena u bazi podataka. Ako u sljedećoj transakciji dođe do greške, ovim pristupom se do zadnjih promjena može doći preko dnevnika, a istovremeno se baza podataka može vratiti u zadnje konzistentno stanje prije greške. Privremena datoteka dnevnika je pohranjena u istom direktoriju kao i datoteka baze podataka. Primjer direktorija je vidljiv na slici 2. Odnosi se na aplikaciju *Skype* za koju je SQLite baza podataka pohranjena u datoteci *main.db*, a pripadni dnevnik je pohranjen u privremenoj datoteci *main.db-journal*.

Slika 2. Izgled SQLite direktorija za aplikaciju *Skype*
Izvor: [2]

Informacija iz zaglavlja da se koristi povratni dnevnik dovoljna je da alat za forenzičku analizu procesira stranice baze podataka. Forenzička analiza se obavlja uz pomoć informacija o stanju baze podataka dobivenih od forenzičkog alata. Prema [2] baza podataka u određenom trenutku je u jednom od sljedećih stanja:

- Ako ne postoji datoteka povratnog dnevnika, onda datoteka baze podataka *main.db* predstavlja zadnje konzistentno stanje.
- Ako datoteka povratnog dnevnika ne sadrži niti jednu stranicu ili ako je nevažeća, onda isto *main.db* predstavlja zadnje konzistentno stanje.
- Ako je datoteka povratnog dnevnika važeća i sadrži jednu ili više stranica baze podataka, onda zadnje konzistentno stanje čine stranice koje se nalaze u povratnom dnevniku u kombinaciji sa svim ostalim stranicama koje su u bazi podataka, ali ne postoje u dnevniku

To je sve što je potrebno znati prije analize rezultata, a svaki današnji alat za forenzičku analizu će automatski odrediti te informacije.

Pristup forenzičkoj analizi se može razlikovati s obzirom na još dva stanja povratnog dnevnika. Prilikom zapljene ili akvizicije uređaja u svrhu provođenja forenzičke analize, često je uređaj uključen. Tada je dnevnik u takozvanom vrućem stanju (*engl. hot state journal*). U tom slučaju je moguće pristupiti dnevniku i bazi i bez naprednih forenzičkih alata. U slučaju isključenog uređaja potrebna je obnova dnevnika forenzičkim alatom i tada se on nalazi u hladnom stanju (*engl. cold state journal*), [2].

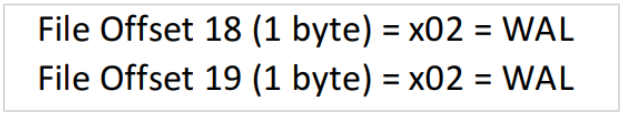
3.1.2. WAL dnevnik

Uporaba povratnog dnevnika je zadani način rada u SQLite-u i u prošlom poglavlju je objašnjeno kako to nije potrebno definirati PRAGMA naredbom. U slučaju WAL dnevnika je potrebno to eksplicitno navesti kako je prikazano u tekstualnom prilogu 2.

```
1.    PRAGMA journal_mode = WAL
```

Tekstualni prilog 2. SQLite naredbe – Definicija uporabe povratnog dnevnika
Izvor: [12]

Isto tako za provjeru korištenog dnevnika može se pristupiti zaglavlju SQLite-a koje se nalazi u korijenu (prvom čvoru) modula B-stabla. U slučaju uporabe WAL dnevnika, dio zapisa koji se referencira na to je dan na slici 3.



Slika 3. SQLite zaglavlje – Povratni dnevnik
Izvor: [13]

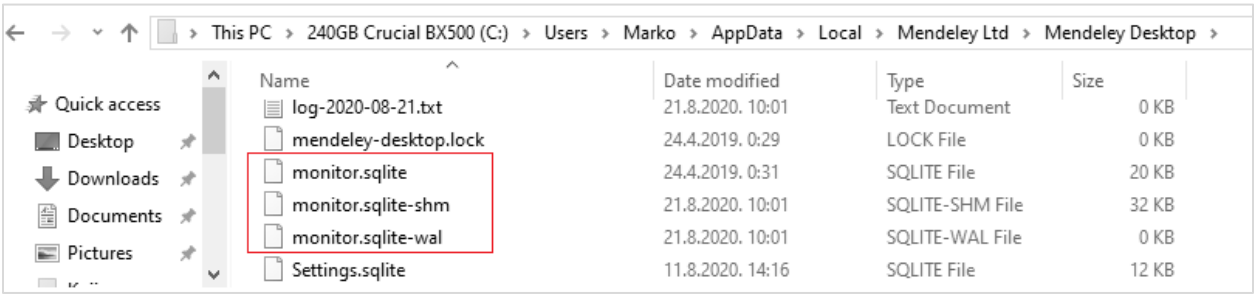
U prethodnom poglavlju su objašnjeni osnovni principi funkcioniranja povratnih dnevnika. Jednostavno rečeno, funkcionira tako što kopiju izvornog nepromijenjenog sadržaja baze podataka piše u zasebnu datoteku povratnog dnevnika, a zatim upisuje promjene izravno u datoteku baze podataka. U slučaju greške ili ROLLBACK naredbe, originalni sadržaj u dnevniku sprema se nazad u datoteku baze podataka. Potvrda svih transakcija se obavlja pri brisanju privremene datoteke dnevnika.

Noviji pristup integritetu baze podataka su WAL dnevnici koji invertiraju pristup problemu integriteta baze podataka. Izvorni sadržaj se čuva u datoteci baze podataka, a sve promjene se dodaju u zasebnu WAL datoteku bez brisanja njezinog prethodnog sadržaja. To omogućuje dodavanje više transakcija u WAL datoteku. Isto tako omogućuje potvrdu transakcija direktno u WAL datoteci bez spremanja u bazu podataka što nije slučaj s povratnim dnevnikom. Prednost je brzina. Zahtjeva se samo jedno zapisivanje podataka za više transakcija koje su potvrđene u WAL datoteci, dok se kod povratnih dnevnika podaci zapisuju za svaku transakciju, [14].

Konkretno, zapis iz WAL datoteke u bazu se događa u kontrolnoj točki koja nastaje na nekoliko načina. Prvi način je zadan automatski i definira nastajanje kontrolne točke u trenutku kad WAL datoteka sadrži tisuću stranica. Drugi način je korisnički definirana kontrolna točka. Korisniku je dopušteno definiranje kontrolne točke naredbama nakon svake transakcije. Treći način nastajanja kontrolne točke je pri zatvaranju programa koji pristupa bazi podataka, [2].

Kada se u WAL dnevniku nalazi više transakcija može se odrediti stanje baze podataka u različitim točkama u vremenu. U većini slučajeva je moguće i odrediti vremenski period u kojemu je podatak izbrisan pregledom zasebnih transakcija.

Slično kao i kod povratnog dnevnika, WAL datoteka je pohranjena u isti direktorij kao i glavna datoteka SQLite baze podataka. Na slici 4 je primjer takvog direktorija za aplikaciju *Mendeley*. Radi se o menadžeru referenci koji pomaže pri skupljanju referenci, organiziranju citata i stvaranju bibliografija.



Slika 4. Izgled SQLite direktorija za aplikaciju *Mendeley*
Izvor: Autor (osobni primjer)

Na slici se primjećuje jedna dodatna datoteka s ekstenzijom *sqlite-shm* (*engl. Shared Memory File*). Svrha te datoteke je traženje mjesta zadnje pojave određene stranice u pripadnom WAL dnevniku. U praksi SHM datoteka sadrži indekse koji se pozivaju prilikom traženja stranica. Isto tako, životni vijek te datoteke je vezan uz dnevnik što znači da se kreira i briše istovremeno kad i WAL datoteka. Početna veličina datoteke je 32 KB, a povećava se za svaku kontrolnu točku u dnevniku i to za 8 B, [2]. Na slici je veličina WAL dnevnika 0 KB što znači da ne postoje transakcije koje trebaju biti zapisane u bazu podataka, a to potvrđuje gornju tvrdnju o početnoj veličini SHM datoteke. Važno je znati da ne postoji forenzička vrijednost SHM datoteke zato što forenzički alati čitaju direktno WAL dnevnik i bazu te nisu potrebni indeksi za tu radnju.

Kada se vrši forenzički analiza nad uređajem koji koristi WAL dnevnik kao način rada SQLite-a, puno je veća vjerojatnost pronalaska transakcija u dnevniku. Razlog tome je gore naveden način spremanja više transakcija u dnevnik. Čest je slučaj pregleda više aplikacija od kojih neke koriste stariji povratni dnevnik, a neke WAL dnevnik. U slučaju zajedničkog pristupanja bazi podataka na višoj razini uređaja, za svaku aplikaciju se kreira WAL dnevnik koji ju povezuje s tom bazom na višoj razini. Mnogo veća je mogućnost pronalaska WAL dnevnika nego povratnih dnevnika, [2].

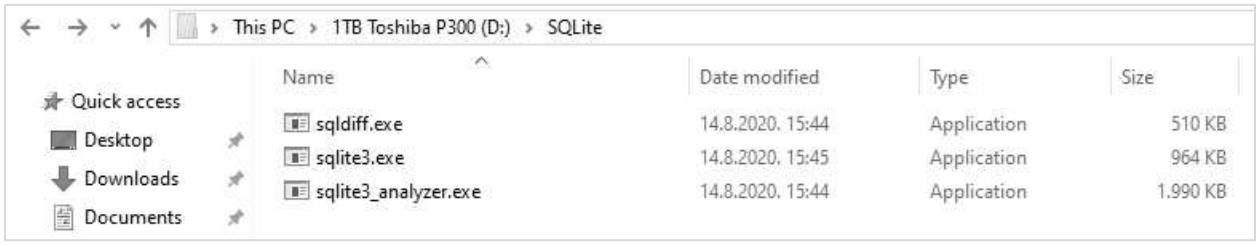
Iz navedenog se zaključuju razlozi raširenosti WAL dnevnika u odnosu na povratne dnevnike. Osim brzine i fleksibilnosti uporabe, ističe se i set podataka. Oba dnevnika sadržavaju isti set podataka, ali WAL datoteka količinski sadrži mnogo više stranica podataka u određenom trenutku.

3.2. SQLite alati

Provedba postupka forenzičke analize za rezultat daje skup podataka koje je nakon analize moguće koristiti kao digitalne dokaze. Za većinu današnjih forenzičkih programa i alata postoje preglednici kojima je moguće predstaviti podatke na razumljiviji način. U većini slučajeva je takav prikaz dovoljan. Nedostatak su opcije pregleda iz perspektive baze podataka. Naime, većina preglednika nudi pregled s obzirom na datotečnu hijerarhiju i vrstu podataka. Ako se želi pogledati struktura baze podataka, potrebno je locirati datoteke baze (obično *.db* ili *.csv* ekstenzije) te ih pregledati SQLite preglednikom ili SQLite komandnim sučeljem. U ovom poglavlju su dane osnove uporabe SQLite-a. Platforma na kojoj se vrši analiza i pregled je *Windows*.

3.2.1. Instalacija i pokretanje SQLite alata

Prvi način pristupa SQLite bazi podataka je putem naredbenog retka. Prvi korak je preuzimanje programskog paketa sa službenih stranica, [15]. Preuzeti paket je komprimiran te ga je potrebno raspakirati u direktorij po želji. Nakon toga odabrani direktorij izgleda kao što je prikazano na slici 5.



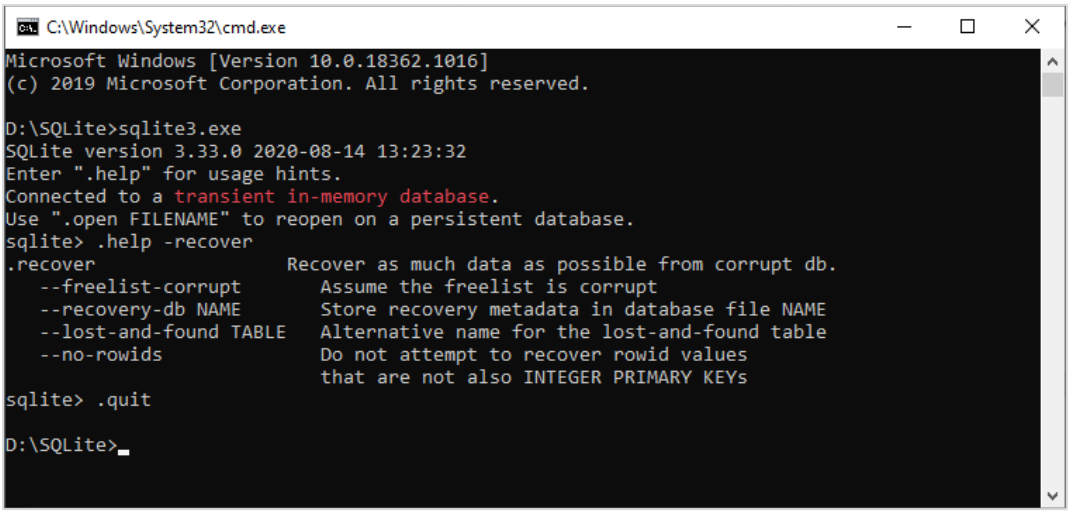
Slika 5. Direktorij iz kojeg se pokreće SQLite alat
Izvor: Autor (osobni primjer)

SQLite se pokreće putem izvršne datoteke *sqlite3.exe* gdje broj tri označava verziju alata. Ostale dvije datoteke služe kao podrška alatu te se ne pokreću zasebno. Osnovni koraci za pokretanje alata, pomoć te izlaz su napisani u tekstualnom prilogu 3.

```
1.    #Pokretanje alata
2.    sqlite3.exe
3.
4.    #Popis dostupnih naredbi
5.    .help
6.
7.    #Izlaz iz alata
8.    .quit
```

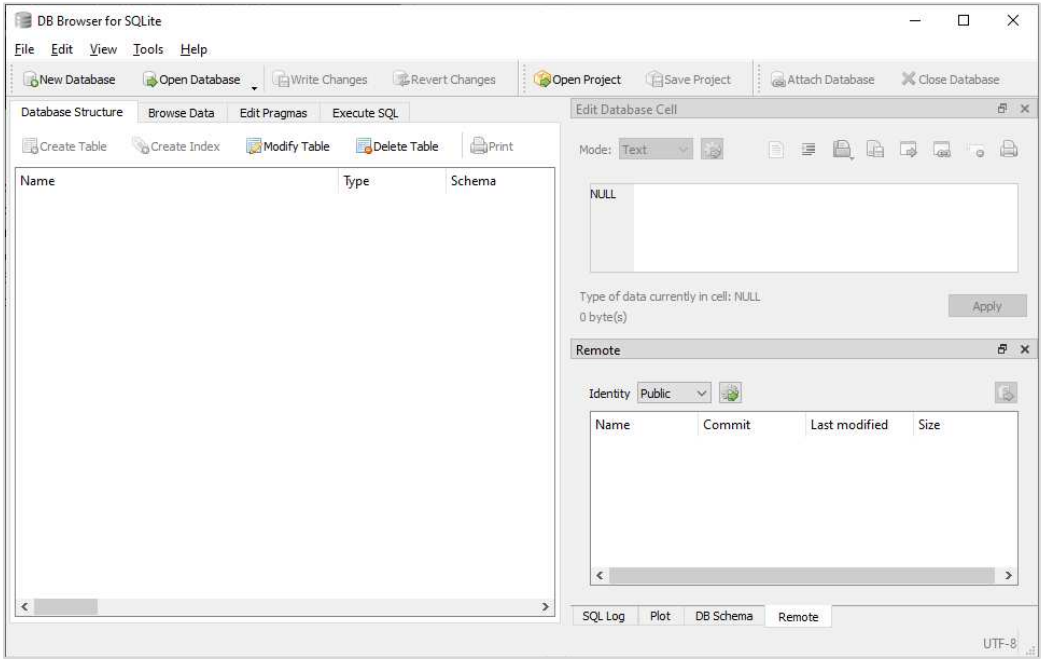
Tekstualni prilog 3. Osnovne naredbe za rad sa SQLite alatom
Izvor: [16]

Ako je sve u redu, povratne informacije naredbenog retka će izgledati kao na slici 6.



Slika 6. Naredbeni redak – osnovne naredbe za rad s SQLite alatom
Izvor: Autor (osobni primjer)

Ovaj način je dovoljan za sve manipulacije nad bazom podataka. Ipak, jednostavnije je raditi sa SQLite bazom podataka pomoću intuitivnog GUI (*engl. Graphical User Interface*) alata. Na raspolaganju su brojni GUI alati od slobodnih do komercijalnih licenci. Najpoznatiji alati slobodne licence su *SQLiteStudio* [17], *DBeaver* [18] i *DB Browser for SQLite* [19] koji će biti korišten u sklopu diplomskog rada u kasnijim poglavljima. *DB Browser* instalacija je jednostavna putem izvršne datoteke dostupe na službenoj stranici, a izgled glavnog sučelja se nalazi na slici 7.



Slika 7. *DB Browser for SQLite* sučelje
Izvor: Autor (osobni primjer)

3.2.2. Uvoz postojećih podataka u tablicu SQLite baze podataka

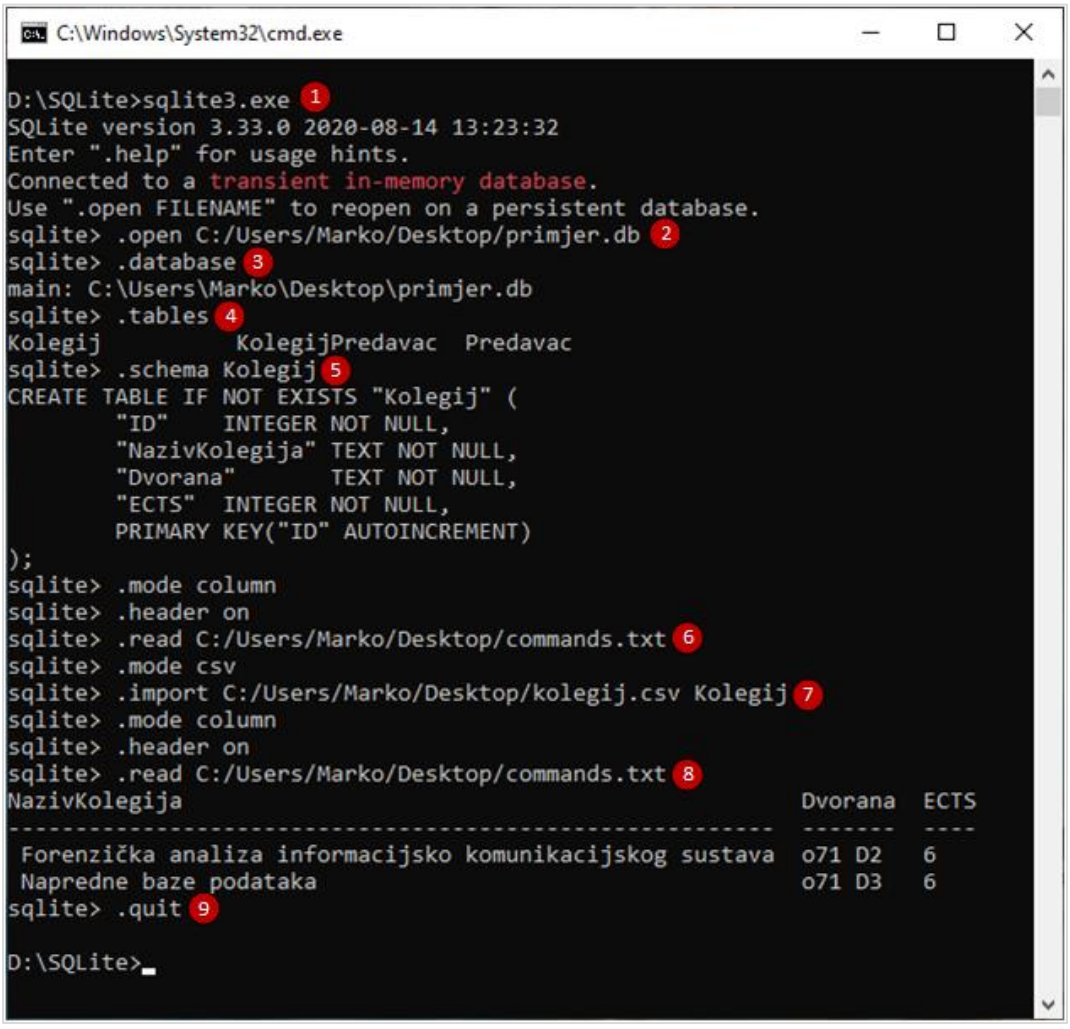
Svi GUI alati nude funkciju uvoza podataka koja omogućuje odabir datoteke koja je u jednom od podržanih formata (npr. *csv*, *json*, *db*). Konkretno u slučaju *DB Browsera* se odabire *File* padajući izbornik i potom opcija *Import database from file*. U slučaju naredbenog retka postoje naredbe kojima se ostvaruje isto. U nastavku poglavlja je objašnjen tijek naredbi za uvoz podataka na taj način.

Prvi korak je, kao i u svim slučajevima, pokretanje alata kako je objašnjeno u tekstualnom prilogu 3. Nakon pokretanja važno je obratiti pažnju na povratni ispis naredbenog retka (slika 6), posebno na redak koji je dijelom ispisan crvenom bojom (*Connected to a transient in-memory database*). To znači da se trenutna SQLite sesija pohranjuje u privremenu memoriju te da će se povijest provedenih naredbi izbrisati kad se sesija zatvori. Sve promjene koje se dogode u samoj bazi ostaju spremljene i nakon sesije, [16]. U pravilu se nikakve promjene ne rade prilikom analize podataka nakon ekstrakcije nego samo pregled. U nastavku je dano objašnjenje naredbi koje će se izvesti u SQLite alatu u ovom poglavlju. Naredbe su vidljive unutar tekstualnog priloga 4.

```
1.  .open FILENAME --otvaranje datoteke baze podataka u sesiji
2.
3.
4.  .mode column   --format ispisa
5.  .header on
6.
7.  .database      --prikaz svih otvorenih baza podataka
8.
9.  .tables        --prikaz svih tablica u trenutnoj bazi
10.                --podataka
11.
12. .schema TABLE  --prikaz strukture tablice
13.
14. .read commands.txt  --izvršavanje upita pohranjenog u
15.                     --tekstualnoj datoteci
16.
17. .mode csv      --uvoz podataka iz csv datoteke
18. .import FILENAME.csv
19.
20.
21.       --Primjer sadržaja datoteke commands.txt
22. (SELECT NazivKolegija, Dvorana, ECTS FROM Kolegij)
```

Tekstualni prilog 4. SQLite naredbe potrebne za primjer
Izvor: [16]

Za primjer je uzeta jednostavna SQLite baza podataka kreirana za potrebe objašnjavanja relacijskog modela. Shema baze podataka se nalazi na grafičkom prikazu 2. Trenutno je baza podataka prazna. Na slici 6 se nalaze koraci za uvoz podataka te njihov pregled.



Slika 8. Postupak uvoza podataka iz CSV datoteke
Izvor: Autor (osobni primjer)

Postavlja se logično pitanje prednosti i nedostataka u odnosu na GUI alate. Na prvu se čini da je GUI alat brži u sličnim postupcima zbog intuitivnosti. Koraci sa slike 6 se mogu obaviti sličnim bojem klikova u sučelju odabranog alata. Ipak, pokazalo se kako je stručnjacima uz malo prakse jednostavnije i brže obaviti posao putem naredbenog retka zato što se sve nalazi unutar jednog ekrana. U nekim okružjima nisu dostupni GUI alati , a to je još jedan razlog za poticanje učenja SQLite naredbi.

Prvim korakom je pokrenut SQLite alat u naredbenom retku. Naredbom pod brojem dva se otvara baza podataka u trenutnoj sesiji. Za podsjetnik, radi se o praznoj SQLite bazi iz poglavlja 2.1. koja je u ovom primjeru pohranjena na radnoj površini u datoteci *primjer.db*.

Naredbom *.database* se dobiva potvrda da je stvarno otvorena odabrana baza podataka. Sljedeća naredba (*.tables*) ispisuje sve entitete u bazi. U primjeru su to tri tablice koje su shemom vidljive na grafičkom prikazu 2. Ako se želi vidjeti shema pojedine tablice u naredbenom retku dovoljno je pokrenuti naredbu pod rednim brojem pet (*.schema*).

Šestim korakom je provedeno čitanje sadržaja tablice Kolegij. Konkretnije, pozvane su naredbe koje su zapisane u datoteci *commands.txt* koja se nalazi na radnoj površini. Sadržaj te datoteke je samo jedna naredba koja odabire sve podatke iz tablice Kolegij, a izgled te naredbe je vidljiv u zadnjem retku tekstualnog priloga 4.

Naredbom se nije ništa ispisalo što potvrđuje početnu tvrdnju da je baza podataka prazna. U međuvremenu je proveden postupak forenzičke analize udaljenog sustava na kojem se nalazi ažurirano stanje ove baze podataka. Ekstrakcijom je dobivena CSV (*engl. Comma Separated Value*) datoteka. Sljedećim korakom se postavlja ispravan mod čitanja te se uvoze podaci iz ekstrahirane datoteke koja se također nalazi na radnoj površini. Naredbom pod brojem osam se ponavlja čitanje tablice Kolegij koja sad sadrži dva uvezena zapisa. Na kraju se zatvara sesija te se izlazi iz SQLite alata.

Pregled podataka u nastavku rada će biti putem prilagođenih grafičkih sučelja. Od prednosti naredbenog retka koje nisu već navedene ističe se dubina pristupa problemu. Naime, čak i najjednostavniji primjer poput ovoga sa slike 8 daje bolji uvid u pozadinu događanja prilikom analize podataka.

4. Ekstrakcija SQLite baze podataka Android uređaja

Prema [5], danas više od 50% svjetske populacije koristi pametni mobilni uređaj. Svjedoči se korisničkoj migraciji s kućnih računala na mobilne telefone. Kako se nove značajke i aplikacije razvijaju, tako i količina podataka pohranjenih na uređajima brzo raste. Mobilni telefoni postaju prijenosni nosači podataka čime prate većinu pokreta i navika korisnika. Uz sve veću prisutnost tih uređaja u svakodnevnom životu, podaci dobiveni iz njih postaju neprocjenjiv izvor dokaza za istrage koje se odnose na građanske, kaznene, pa čak i visokoprofilne slučajeve. Danas su rijetki postupci provedbe forenzičke analize koji ne uključuju pametni mobilni telefon.

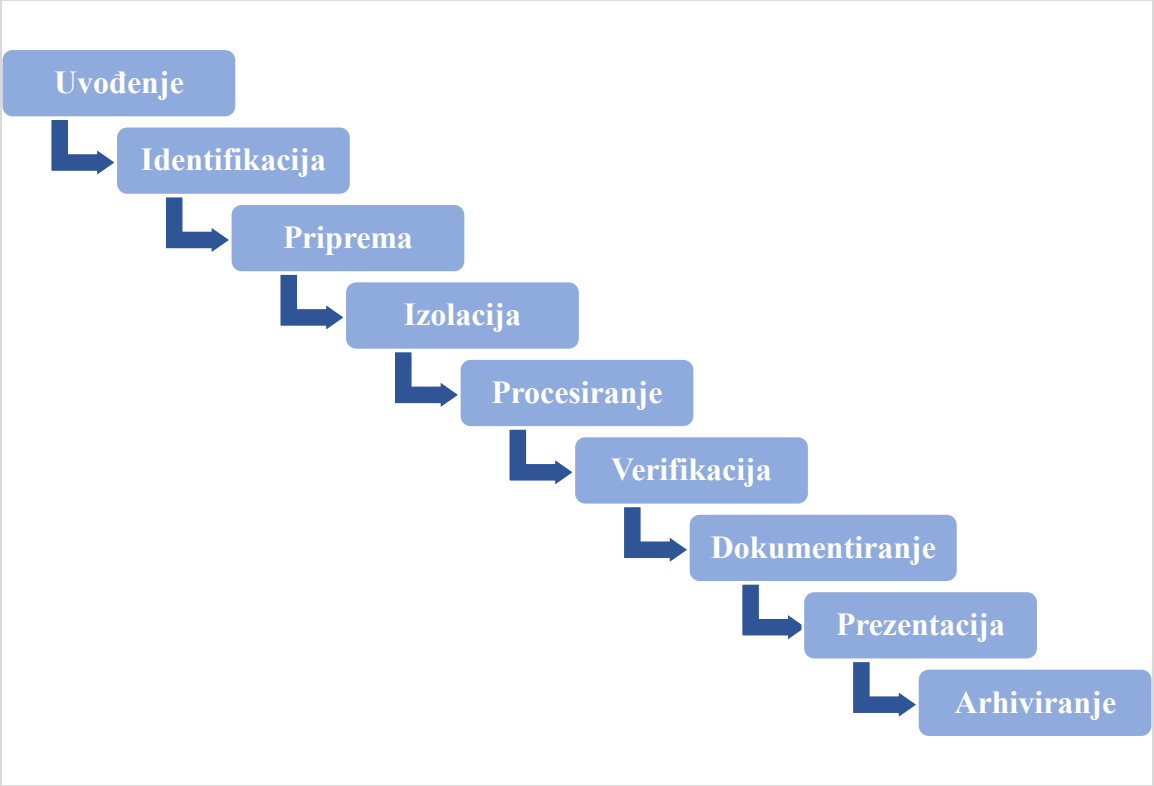
Digitalna forenzika grana je forenzičke znanosti koja je usredotočena na oporavak i istragu podataka koji nalaze u elektroničkim ili digitalnim uređajima. Postoje razne grane digitalne forenzike na temelju vrste uključenih digitalnih uređaja, a najraširenije su računalna, mrežna i mobilna forenzika. Proces mobilne forenzike se može podijeliti u tri osnovne kategorije, a to su zapljena ili stjecanje uređaja, akvizicija ili ekstrakcija podataka te analiza podataka. U ovom poglavlju je cijeli proces detaljnije objašnjen. Isto tako, definirane su razlike pristupa ekstrakciji podataka i dan je primjer ekstrakcije odabranim alatom.

4.1. Postupak ekstrakcije podataka mobilnih uređaja

Forenzičko ispitivanje svakog mobilnog uređaja može se razlikovati. Praćenjem definiranih koraka u procesu forenzičke analize osigurava se da su podaci i proces dobro dokumentirani te da je postupak ponovljiv i dokaziv. Ne postoji u potpunosti standardni postupak za sve uređaje, ali je definirana referentna metodologija koja se nalazi na grafičkom prikazu 5. Svi koraci moraju biti testirani, potvrđeni i dokumentirani.

Početna faza zahtjeva **uvoz podataka**. Standardni dokumenti ovog koraka su obrazac o vlasništvu uređaja, dokument opisa vrste incidenta u kojem je uređaj sudjelovao (ili drugi razlog pokretanja postupka) te okvirna definicija podataka i informacija koje se traže. Definiranje ciljeva forenzičke istrage je ključno u ovom koraku. Time se pojednostavljuju zadaci koje mora ostvariti ispitivač. Naravno, prilikom akvizicije uređaja je potrebno paziti da se ne modificira niti jedan pohranjen podatak, [20].

Sljedeći korak je **identifikacija**. Tu je potrebno odrediti osnovne detalje za daljnji postupak. Prvo se određuje pravno ovlaštenje čime se definiraju ograničenja za ispitivanje uređaja. Na primjer, ako se ispitivanje vrši na temelju naloga, onda treba sukladno ograničiti područje pretrage. Nakon toga identificiraju se model i vezani podaci o uređaju. Kao i svakom koraku, sve je potrebno dokumentirati.



Grafički prikaz 5. Referentna metodologija ekstrakcije podataka mobilnog telefona
Izvor: [21]

Kada se identificiraju svi parametri, prelazi se u fazu **pripreme**. Faza uključuje istraživanje metoda i alata kojima će se prikupiti podaci. Osim modela uređaja, odabir alata se temelji na više faktora poput cilja i opsega ispitivanja, raspoloživih resursa te prisutnosti vanjske pohrane, [20].

Izolaciju uređaja je potrebno obaviti što ranije. Prilikom akvizicije uređaja je dobra praksa pohraniti ga u *Faraday-evu* vrećicu kako bi se onemogućile komunikacijske sposobnosti koje mogu utjecati na podatke. Dobra praksa je postavljanje uređaja u zrakoplovni način rada ako ispitivač ima pristup. Za jednostavniji rad s uređajem danas se konstruiraju *Faraday-evi* šatori i sobe koje također blokiraju svu komunikaciju s vanjskim mrežama, [20].

Jednom kada je uređaj u potpunosti izoliran, započinje **procesiranje**. Preferira se metoda fizičke ekstrakcije direktno iz memorije. Time se dobivaju takozvani sirovi podaci. Uređaj je tijekom te metode isključen te se zbog toga ne događaju nikakve promjene u memoriji. Zato je to preferirana metoda. Naravno, fizička ekstrakcija nije uvijek moguća. Tada se pristupa drugim metodama koje su detaljnije objašnjene u sljedećem poglavlju.

Nakon procesiranja je potrebno **verificirati** točnost dobivenih podataka kako bi se osiguralo da nije došlo do izmjena. Često se za verificiranje koristi više alata za ekstrakciju podataka nakon čega se uspoređuju rezultati, [22].

Forenzički ispitivač je dužan je dokumentirati cijeli postupak. Iako je dokumentiranje sadržano u svakom koraku, uvedena je posebna faza **dokumentiranje** u kojoj se vrši revizija i uređivanje dokumentacije. Nakon što su ranije faze provedene, mora postojati neki oblik stručnog nadzora koji će na temelju dokumentacije potvrditi da je postupak obavljen unutar definiranih pravila i ograničenja. Standardna dokumentacija, između ostalog sadrži [20]:

- Datum i vrijeme početka postupka
- Opis fizičkog stanja uređaja
- Fotografije uređaja i pojedinih komponenti
- Status uređaja prilikom njegove akvizicije
- Marku i model uređaja
- Opis alata korištenog u postupku
- Podatke dobivene postupkom
- Bilješke nadzornika i recenzenata dokumentacije

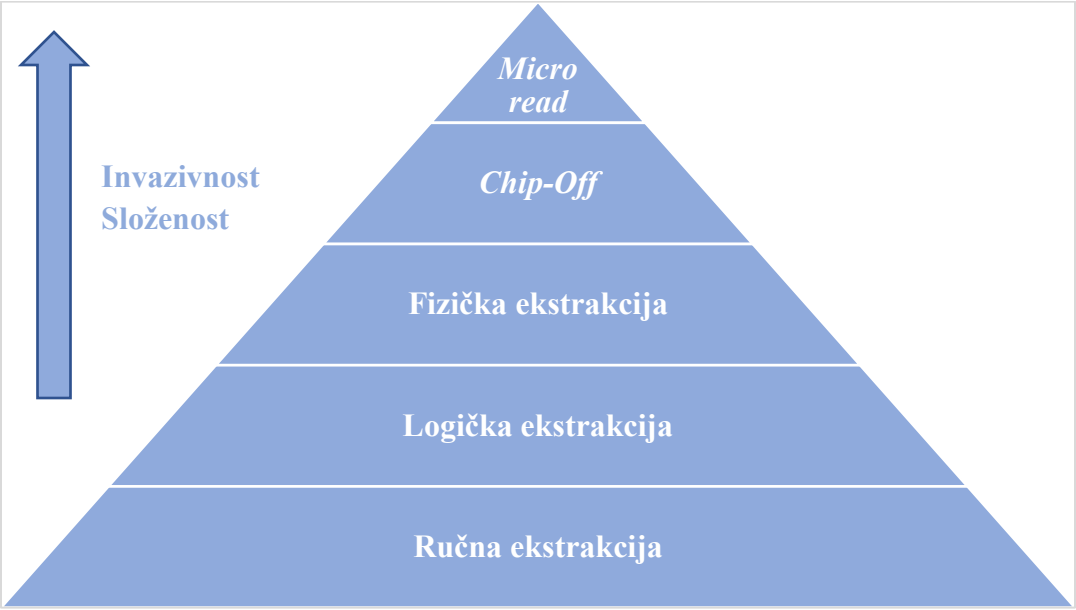
Tijekom stvaranja dokumentacije potrebno je obratiti pažnju na mogućnost **prezentiranja** rezultata. Mora se osigurati mogućnost jasnog i jednostavnog predstavljanja rezultata drugim ispitivačima, sudskim službenicima u slučaju sudskog postupka ili bilo kojim osobama od interesa. Prezentacijska izvješća su danas standardno u elektroničkom formatu, ali nije ih problem prikazati u papirnatom obliku. Razlog tome su standardni obrasci dokumentacije. Podaci sadržani u prezentaciji moraju biti sažeti, jasni i najvažnije ponovljivi. Isto tako, prezentacija mora pratiti vremensku liniju postupka, a to olakšava forenzički alat tako što pri ekstrakciji dodaje vremenski token svakom podatku, [22].

Očuvanje i **arhiviranje** prikupljenih podataka čine zadnju, ali ne i manje važnu fazu. Važno je čuvati podatke u odobrenom formatu za sve buduće reference koje će biti potrebne. Nadalje, potrebno je osigurati zalihost u slučaju oštećenja datoteka. Za svaku kopiju, pohranu ili pristup arhivi se vodi evidencija. Čest je slučaj dugog trajanja sudskih predmeta i postupaka. Ponekad traju više godina pa je jasna važnost postojanja arhiva kako bi se jednostavno moglo pristupiti podacima u bilo kojem trenutku.

Metodologija opisana u ovom poglavlju ne mora u potpunosti odgovarati stanju u stvarnosti. Ona služi kao detaljni set smjernica prilikom provođenja postupka forenzičke analize. Određene faze ne mogu biti provedene prije završetka drugih faza. Neki postupci se provode tijekom cijelog procesa (npr. izolacija i dokumentiranje) iako postoje posebno definirane faze za te postupke. Te sitnice ne utječu na krajnji rezultat, a ovise o specifičnosti slučaja i zakonskoj regulativi. Navedena referentna metodologija prihvaćena je kao uobičajen način provođenja forenzičke analize mobilnih uređaja.

4.2. Metode ekstrakcije podataka mobilnih uređaja

Forenzička analiza mobilnih uređaja uključuje uporabu automatiziranih alata, ali i ručni napor te znanje stručnjaka i ispitivača. Postoji mnoštvo dostupnih alata, a svi imaju određene prednosti i nedostatke. Važno je razumjeti da u nekim postupcima niti jedan alat zasebno nije dovoljan za sve svrhe. Ipak, za potrebe edukacije i jednostavnih primjera, većina alata ima dovoljno funkcionalnosti što čini odabir lakšim. Odabir alata će u nekim slučajevima ovisiti o željenoj razini pristupa uređaju i potrebi za određenim setom podataka koji je teže dostupan. Na grafičkom prikazu 6 se nalazi prikaz metoda ekstrakcije podataka mobilnih uređaja ovisno o razini invazivnosti.



Grafički prikaz 6. Hijerarhijski prikaz metoda ekstrakcije mobilnih uređaja
Izvor: [23]

Cilj ovakve klasifikacije je omogućiti ispitivaču odabir forenzičkog alata na temelju metodologije ekstrakcije. Svaki forenzički alat se može svrstati u jednu ili više prikazanih razina. Počevši od donje razine grafičkog prikaza prema vrhu, alati postaju tehnički složeniji, kompleksniji, invazivniji te zahtijevaju duže vrijeme za ekstrakciju i analizu. Ispitivač mora znati zahtjeve i prema tome odrediti optimalnu metodu ekstrakcije. Ne treba pretjerati s kompleksnošću što zahtjeva previše vremena, a opet se mora postići tražena razina invazivnosti kako bi se prikupili potrebni podaci. Postoji i opasnost od uništavanja dokaza ako se nepravilno odabere alat. Taj rizik se povećava kako se kreće prema vrhu piramide. Zato je za postizanje uspješne ekstrakcije potrebna odgovarajuća edukacija.

Ručna ekstrakcija ne zahtijeva poseban forenzički alat. Ova metoda podrazumijeva jednostavno čitanje podataka direktno s uređaja pomoću tipkovnice ili ekrana na dodir. Otkrivene informacije se dokumentiraju fotografijom. Postupak ekstrakcije je brz i jednostavan za korištenje, a funkcionirat će na gotovo svakom uređaju uz uvjet da mu je dozvoljen pristup. Metoda je sklona ljudskoj pogrešci. Na primjer, mogu se propustiti određeni podaci zbog nepoznavanja sučelja uređaja. Ovom metodom nije moguće oporaviti izbrisane podatke, [20]. Isto tako je vrlo vjerojatno da će doći do promjene podataka što narušava pravilo ekstrakcije. Na primjer, pregled nepročitanog SMS-a (*engl. Short Message Service*) će ga označiti kao pročitanog.

Logička ekstrakcija uključuje povezivanje uređaja s forenzičkim alatom (forenzički softver, hardver ili radna stanica). Kroz forenzički alat se pokreće naredba koja se šalje uređaju i potom interpretira unutar CPU-a (*engl. Central Processing Unit*) uređaja. Naredbom se dohvaćaju podaci iz memorije uređaja i zatim se šalju nazad prema forenzičkom alatu putem kojeg ispitivač može pregledati i analizirati podatke. Većina trenutno dostupnih forenzičkih alata djeluje na ovoj razini ekstrakcije. Postupak je i u ovom slučaju brz, jednostavan i zahtjeva minimalnu edukaciju ispitivača. Loša strana je mogućnost zapisivanja podataka na uređaj prilikom spajanja i slanja naredbi. Uz to, oporavak izbrisanih podataka je moguć u manjini ovakvih alata, [20].

Fizička ekstrakcija je u literaturi poznata još i kao ***Hex Dump*** metoda. Prvi korak je kao i kod prethodne metode spajanje uređaja i forenzičkog alata. Razlika je u tome što fizički forenzički alati šalju poseban kod prema uređaju koji nosi instrukciju za ekstrakciju čitavog memorijskog prostora od čega i dolazi engleski naziv metode. Rezultat je memorijska datoteka u binarnom formatu. Forenzički alat iz datoteke slaže podatke za prikaz, ali ne postoji mogućnost za dolazak do svih podataka na taj način. Potrebna je dodatna analiza za koju je potrebna stručnost ispitivača. Proces je nešto kompliciraniji i zahtjeva višu razinu obuke i edukacije, ali zauzvrat pruža više podataka. Najvažnija karakteristika je mogućnost oporavka izbrisanih podataka zato što je ovdje omogućen pristup cijelom memorijskom prostoru, pa tako i nealociranom dijelu, [22].

Chip-Off se odnosi na prikupljanje podataka izravno s memorijskog čipa. Na ovoj razini, memorijski čip se fizički uklanja s uređaja, a posebni forenzički alat se koristi za ekstrakciju pohranjenih podataka na njemu. Metoda je tehnički mnogo zahtjevnija u odnosu na prethodne metode. Isto tako, proces je skup i zahtjeva znanje na hardverskoj razini uz poznavanje forenzičkog alata. Potrebne vještine uključuju lemljenje i pravilno zagrijavanje memorijskog čipa. Čak i najmanja pogreška u proceduri može učiniti čip neiskoristivim. Ako je moguće, preporuka je pokušati provesti sve ostale manje destruktivne metode ekstrakcije. *Chip-Off* je poželjan u situacijama kada je potrebno sačuvati stanje memorije u točno onakvom obliku u kakvom postoji na uređaju, [20]. Također je jedina opcija kad je uređaj oštećen, ali je memorijski čip sačuvan.

Micro read proces uključuje ručni pregled i interpretaciju podataka s memorijskog čipa. Ispitivač koristi elektronski mikroskop za analizu strukture čipa. Čip je programiran na najnižoj razini digitalne logike uporabom strukture programabilnih vrata. Ispitivač prevodi status tih logičkih vrata u nule i jedinice kako bi odredio niz zapisanih znakova. Cijeli proces je znatno dugotrajniji i kompliciraniji u usporedbi s ostalim metodama i zahtjeva znatno veću razinu znanja. *Micro read* se zbog ekstremnih tehničkih karakteristika gotovo nikada ne koristi, čak i u slučaju nemogućnosti uporabe drugih metoda. Jedni slučajevi koji opravdavaju korištenje ove metode su visokoprofilni slučajevi koju ugrožavaju nacionalnu sigurnost. Posljedica rijetke provedbe ove metode je nepostojanje dovoljno dobre dokumentacije. Također, trenutno ne postoje komercijalno dostupni forenzički alati koji bi olakšali i ubrzali *Micro read*.

4.3. Primjer ekstrakcije podataka Android uređaja

U ovom poglavlju je proveden jednostavan primjer ekstrakcije podataka *Android* uređaja. Radi se o metodi logičke ekstrakcije koja je dovoljna za detaljni prikaz podataka. Fokus je stavljen na prikaz postupka i dobivanje željenih podataka. Navedene su osnovne karakteristike forenzičkog alata i ciljnog uređaja. U ovako jednostavnom primjeru ne postoji potreba za detaljnom procedurom referentne tehnologije. Unatoč tome, svaki korak proveden u poglavlju logički pripada određenoj fazi. Na primjer, dio diplomskog rada se može promatrati kao slobodnija definicija dokumentacije, a instalacija i postavljanje forenzičkog alata podrazumijeva pripremu.

4.3.1. Forenzički alat

Odabran je forenzički alat *Andriller*. Radi se o softveru s kolekcijom forenzičkih alata za pametne telefone. Razvijen je za ekstrakciju podataka *Android* operativnog sustava. Osim ekstrakcije, posjeduje i druge mogućnosti poput određivanja uzorka za zaključavanje zaslona, određivanja lozinki za zaključavanje i dekodiranja SQLite baza podataka za većinu dostupnih aplikacija. Izvješća su generirana u HTML (*engl. Hyper Text Markup Language*) i *Excel* formatima. *Andriller* ima slobodnu licencu te je otvorenog koda. Dostupan je za preuzimanje na službenom *GitHub* repozitoriju, [24]. Preuzima se direktorij koji sadrži sve potrebno za instalaciju i pokretanje GUI sučelja.

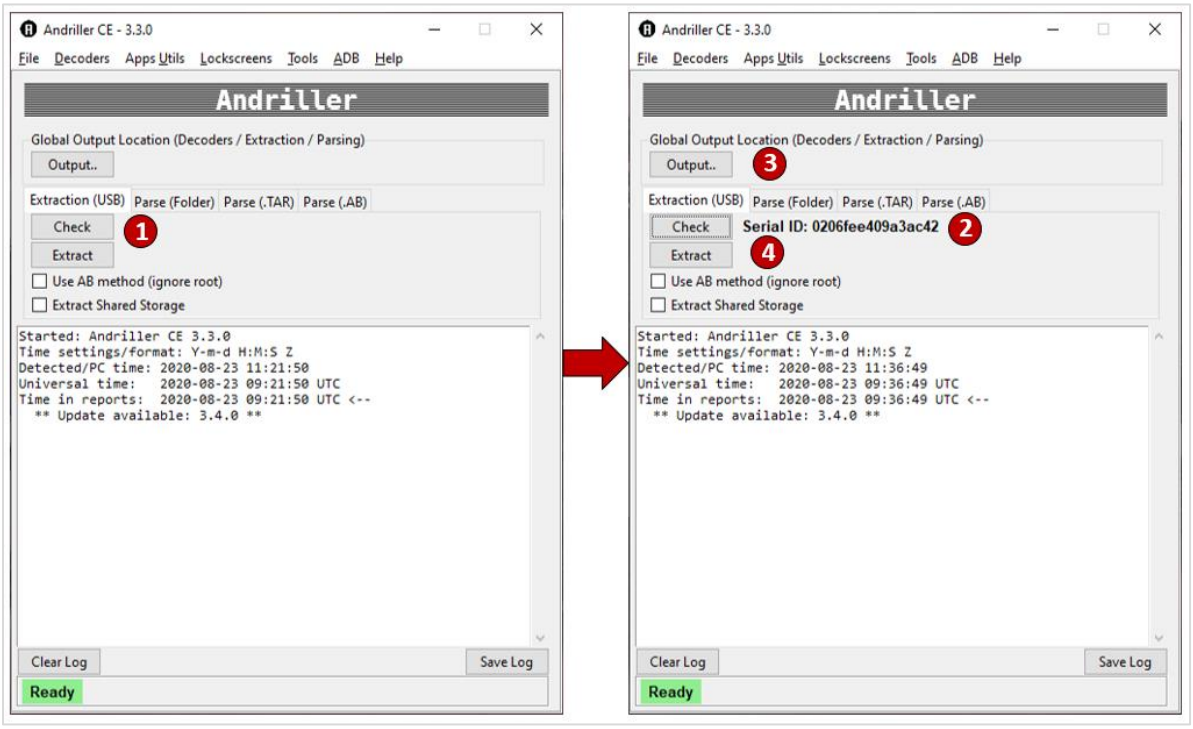
Instalacija nije jednostavna putem izvršne datoteke. Potrebno je pratiti niz uputa dostupnih na stranici za preuzimanje. Preporuka je pokretanje alata unutar virtualnog okruženja. Postupak instalacije i pokretanja sučelja se izvršava putem naredbenog retka na sljedeći način:

1. Instalacija *Python* programskog paketa ako ne postoji na računalu.
2. Kreiranje direktorija koji će služiti kao virtualno okruženje uporabom *Python* naredbe.
3. Pozicioniranje u stvoreni direktorij i aktiviranje virtualnog okruženja putem stvorene aktivacijske skripte.
4. Izvršavanje naredbe za instalaciju *Andriller* alata.
5. Izvršavanje naredbe za pokretanje GUI sučelja.

Naredbe su jednostavnog formata, ne zahtijevaju poznavanje šire problematike virtualnih okruženja te su dostupne na spomenutom *GitHub* repozitoriju, [24]. Iz tog razloga nije potreban detaljniji uvid u strukturu naredbi.

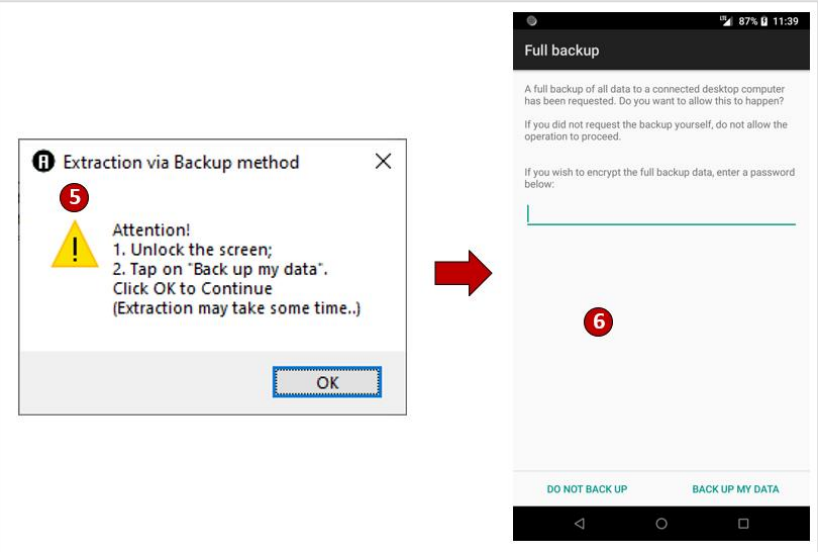
4.3.2. Proces ekstrakcije putem Andriller forenzičkog alata

Nakon uspješne instalacije i pokretanja otvara se GUI sučelje prikazano na slici 9.



Slika 9. Ekstrakcija uporabom *Andriller* alata – prvi dio
Izvor: Autor (osobni primjer)

Naravno, potrebno je povezati *Android* uređaj putem podatkovnog USB (*engl. Universal Serial Bus*) kabla. Korakom jedan na slici 9 se odabire opcija provjere povezanog uređaja. Rezultat vraća serijski identifikator uređaja čime je potvrđeno uspješno povezivanje. Prije ekstrakcije je još potrebno odabrati izlazni direktorij za pohranu podataka. To je označeno korakom tri. Nakon definiranja navedenih parametara potrebno je pokrenuti ekstrakciju.



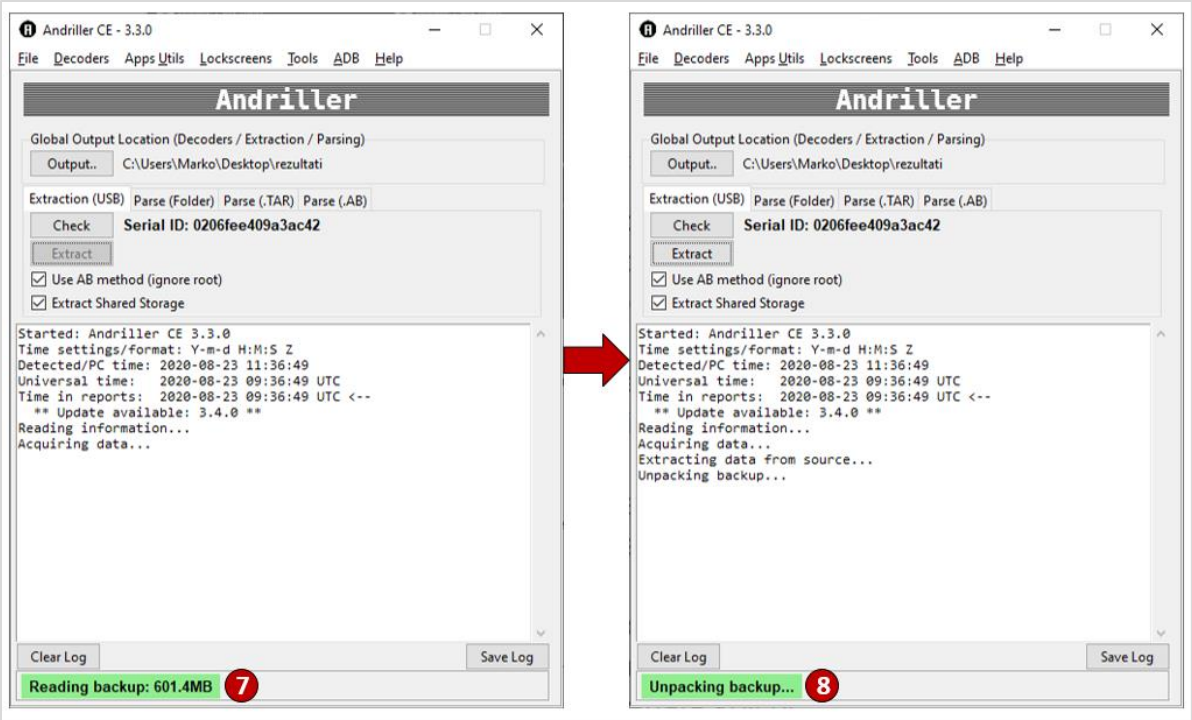
Slika 10. Ekstrakcija uporabom *Andriller* alata – drugi dio
Izvor: Autor (osobni primjer)

Time se pokreće skočni prozor koji je prikazan na slici 10. Potrebno je odabrati opciju OK. U isto vrijeme se otvara novi ekran na uređaju koji se također nalazi na slici 10. Unutar prozora je potrebno potvrditi kreiranje sigurnosne kopije podataka.

Nakon svih potrebnih odobrenja pokreće se proces ekstrakcije podataka. Unutar glavnog prozora se kreiraju zapisi koji prate proces.

Prvo se obavlja kreiranje sigurnosne kopije. Ovaj korak je vremenski najopsežniji i može potrajati više sati. Ovisi o veličini podataka na *Android* uređaju, broju instaliranih aplikacija, procesorskoj moći računala i ograničenjima podatkovnog kabla. Sigurnosna kopija je u formatu s ekstenzijom *.ab*.

Na kraju se obavlja postupak raspakiravanja (izdvajanja datoteka) sigurnosne kopije čime se automatski kreira izvještaj. Nastaje mnogo direktorija koji sadržavaju podatke aplikacija i uređaja.



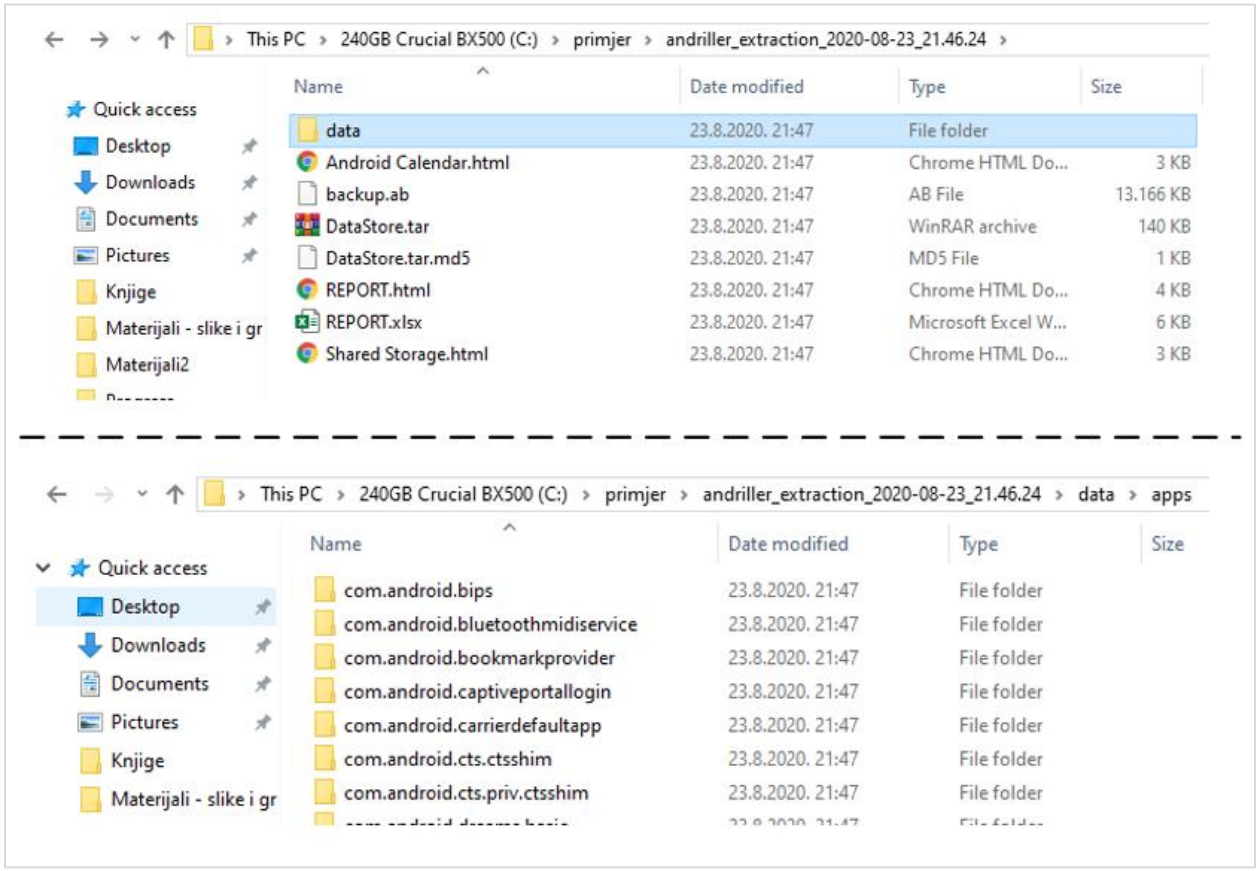
Slika 11. Ekstrakcija uporabom *Andriller* alata – treći dio
Izvor: Autor (osobni primjer)

U donjem lijevom kutu je ispisan trenutni korak procesa što je označeno na slici 11. Korakom sedam je označeno kreiranje sigurnosne kopije gdje je dostupna informacija o trenutnoj veličini datoteke u koju se kopija pohranjuje.

Zadnjim korakom koji je označen brojem osam nastaju direktoriji koji su prikazani u sljedećem poglavlju. Tako nastali direktoriji omogućuju analizu podataka koja ne bi bila moguća pomoću izvorne datoteke sigurnosne kopije uređaja.

5. Analiza Android baze podataka na razini uređaja i aplikacija

Proces ekstrakcije nudi podatke za svaku aplikaciju, ali kreira i izvještaj na razini uređaja. Podaci aplikacija su raspoređeni u zasebnim direktorijima u kojima se nalaze i SQLite datoteke (*.db*) pomoću kojih će se analizirati podaci. Dobiveni direktorij je prikazan na slici 12.



Slika 12. Direktorij dobiven ekstrakcijom podataka
Izvor: Autor (osobni primjer)

U glavnom direktoriju se primjećuje datoteka izvješća. Ta datoteka se obično prva dokumentira te se iz nje dobivaju osnovni podaci na razini uređaja. Format datoteke je HTML pa se izvješće vrlo lako može otvoriti u *web* pregledniku. *Andriller* kreira izvješće u tabličnom obliku, a prevedeni sadržaj se može vidjeti u tablici 1 uz napomenu da su određeni dijelovi uskraćeni od strane autora zbog zaštite osobnih podataka.

Tablica 1. *Andriller* izvješće na razini uređaja

Tip	Podaci
Serijski identifikator	0206fee409a3ac42
Status	uređaj
Dopuštenje	*shell*
Model	Redmi Note 5
IMEI	**********
Android verzija	7.1.2
Wifi MAC	20:4*:**:**:*2:4a
Lokalno vrijeme	2020-08-23 11:36:23 Central European Daylight Time
Uređaj vrijeme	2020-08-23 11:36:23 CEST
Računi	• com.google: marko********mail.com • com.whatsapp: WhatsApp • com.viber.voip: +385********1

Izvor: Autor (osobni primjer)

Tijekom forenzičke istrage, često nije dovoljno analizirati podatke na razini uređaja. Za razliku od starijih uređaja, danas se većina podataka veže uz aplikacije. To je slučaj čak i kod podataka koji su logički vezani više uz uređaj. Na primjer, zapisi o pozivima ili SMS porukama su osnovni podaci uređaja, ali imaju zasebne aplikacije.

Također, aplikacije kao što su *WhatsApp*, *Gmail i Google Maps* sadrže podatke koji su u većini slučajeva od veće važnosti od osnovnih podataka na razini uređaja. Stoga je važno proučiti lokaciju pohrane tih podataka. Osim lokacije, važno je razumjeti strukturu pohrane. Na primjer, jedna aplikacija može imati više datoteka SQLite baze podataka zbog potrebe za pohranom različitih skupova podataka. Jedan skup se može odnositi na korisnika uređaja, a drugi recimo na metapodatke koji definiraju vremenski okvir, [25]. To znanje pomaže u pronalasku SQLite datoteka pomoću kojih se čitaju podaci u preglednom tabličnom obliku.

Odabrani SQLite GUI alat u diplomskom radu je *DB Browser for SQLite*.

5.1. Google Chrome analiza

Google Chrome je jedan od najzastupljenijih *web* preglednika na današnjim *Android* uređajima. Podaci aplikacije se mogu pronaći u direktoriju *com.android.chrome* i njegovim pod-direktorijima. Prema [20], u nastavku su navedeni opisi i lokacije najvažnijih setova podataka za *Google Chrome*:

- **Oznake** → Tu su sadržani podaci o svim spremljenim oznakama preglednika koje se referenciraju na *web* stranice. Neki od podataka su ime stranice, poveznica do stranice i vremenski token kad je oznaka spremljena. Put do oznaka je sljedeći: *com.android.chrome/app_chrome/Default/Bookmarks.*
- **Podaci za prijavu** → Svi računi za prijavu za koje je definirano da se pohranjuju u *Google Chrome* nalaze se u ovoj datoteci. Tako ova baza podataka uz ostalo sadrži korisnička imena, lozinke i poveznice do stranica. Pristupiti joj se može na lokaciji *com.android.chrome/app_chrome/Default/LoginData.*
- **Ostali podaci** → Radi se o podacima poput telefonskih brojeva ili adrese e-pošte koje korisnik unese tijekom ispunjavanja obrasca na različitim web mjestima. Ti podaci se najčešće koriste za automatsko ispunjavanje formi, a dostupni su na *com.android.chrome/app_chrome/Default/WebData.*
- **Povijest pregledavanja** → Više tablica koje su vezane uz povijest pregledavanja su pohranjene u SQLite datoteku *History.db* koja se nalazi na sljedećoj lokaciji: *com.android.chrome/app_chrome/Default/History.*

Upravo je povijest pregledavanja najzanimljivija stavka u analizi i zato je na slici 13 prikazana jedna od tablica unutar *History.db* SQLite baze podataka. Radi se o tablici koja pohranjuje pretraživane termine te vezane vremenske oznake. Baza podataka je otvorena alatom *DB Browser for SQLite*.

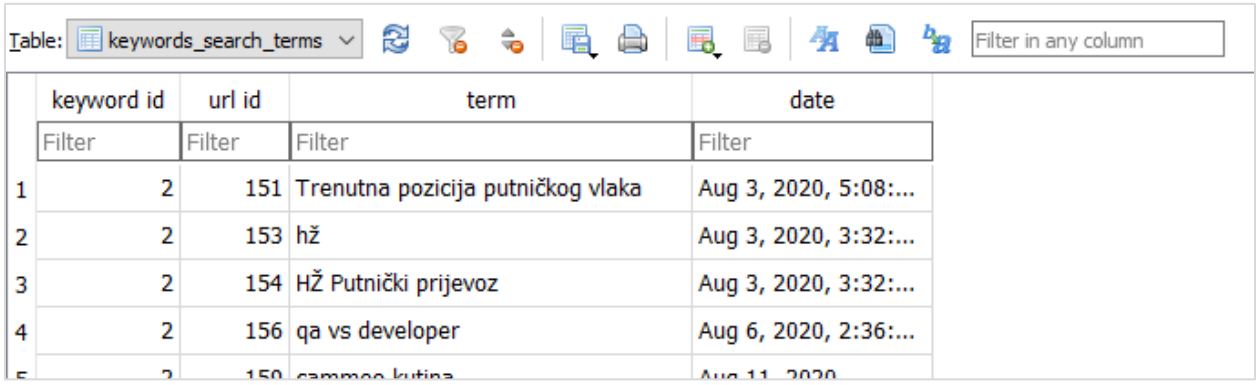


Table: keywords_search_terms — Filter in any column

	keyword id	url id	term	date
	Filter	Filter	Filter	Filter
1	2	151	Trenutna pozicija putničkog vlaka	Aug 3, 2020, 5:08:...
2	2	153	hž	Aug 3, 2020, 3:32:...
3	2	154	HŽ Putnički prijevoz	Aug 3, 2020, 3:32:...
4	2	156	qa vs developer	Aug 6, 2020, 2:36:...

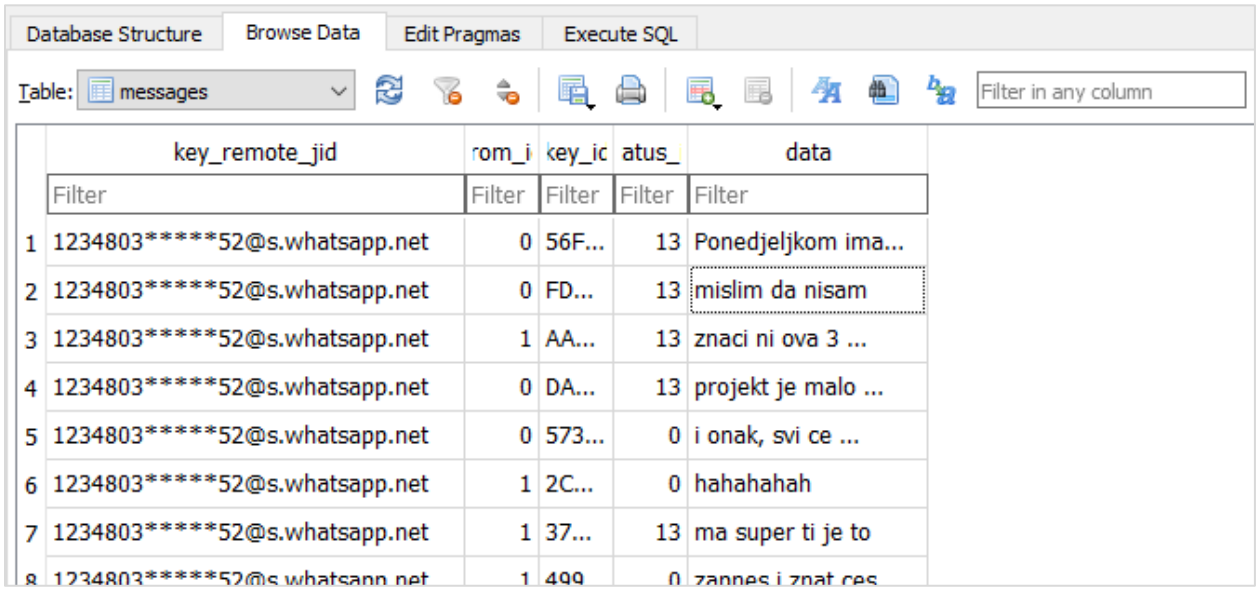
Slika 13. Tablica *keyword_search_terms* iz *History.db* SQLite baze podataka
Izvor: Autor (osobni primjer)

5.2 WhatsApp analiza

WhatsApp je najpopularnija usluga razmjene poruka te audio i video snimki u direktnoj komunikaciji. Broj korisnika na mjesečnoj bazi prelazi 1.5 milijardi, [26]. Također postoji poseban direktorij za *WhatsApp*, a on je *com.whatsapp*. U ovom slučaju su zanimljivi sljedeći setovi podataka, [20]:

- **Korisnička profilna slika** → Radi se o jednostavnoj JPG (*engl. Joint Photographic Experts Group*) datoteci točnog naziva *me.jpg*. Ovo je odličan primjer podatka koji je u pogodnom obliku za pregled bez uporabe alata. Datoteka se nalazi u početnom direktoriju aplikacije.
- **Medijske datoteke** → Još jedan primjer datoteka koje se mogu ručno analizirati bez upotrebe alata. Radi se o svim datotekama koje su dijeljene s aplikacijom poput slikovnih, audio i video zapisa. Zanimljivost je lokacija pohrane koja nije vezana uz *com.whatsapp* nego se radi o posebnoj lokaciji *.../ WhatsApp/Media*.
- **Poruke** → Ključni set podataka za analizu *WhatsAppa* pohranjen u SQLite bazu podataka pod imenom *msgstore.db* na lokaciji *com.whatsapp/databases*.

Na slici 14 je otvorena *msgstore.db* SQLite baza podataka i dan je pregled najzanimljivije tablice imena *messages* u kojoj se između ostalog nalazi sadržaj poruka.



	key_remote_jid	rom_i	key_ic	atus_	data
	Filter	Filter	Filter	Filter	Filter
1	1234803*****52@s.whatsapp.net	0	56F...	13	Ponedjeljkom ima...
2	1234803*****52@s.whatsapp.net	0	FD...	13	mislim da nisam
3	1234803*****52@s.whatsapp.net	1	AA...	13	znaci ni ova 3 ...
4	1234803*****52@s.whatsapp.net	0	DA...	13	projekt je malo ...
5	1234803*****52@s.whatsapp.net	0	573...	0	i onak, svi ce ...
6	1234803*****52@s.whatsapp.net	1	2C...	0	hahahahah
7	1234803*****52@s.whatsapp.net	1	37...	13	ma super ti je to
8	1234803*****52@s.whatsapp.net	1	499	0	zapnes i znat ces

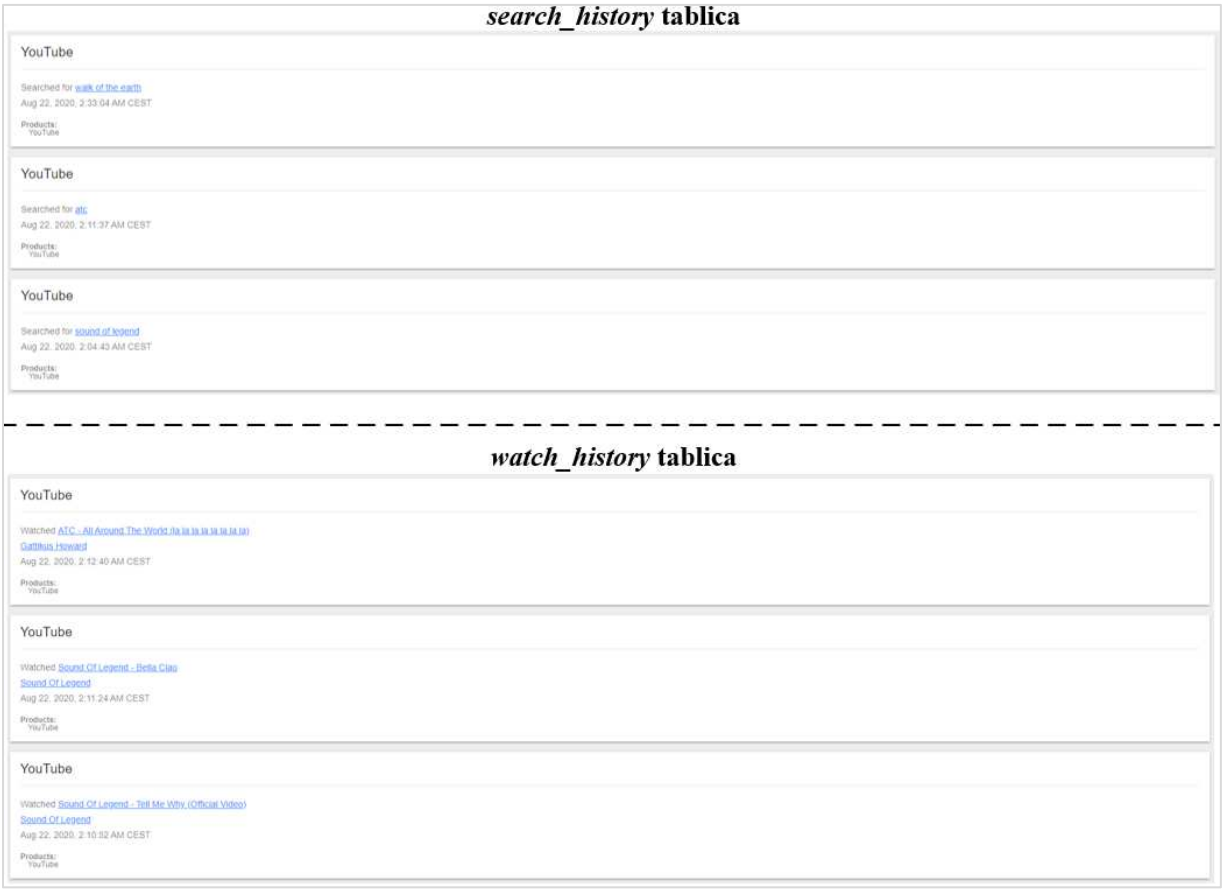
Slika 14. Tablica *messages* iz *msgstore.db* SQLite baze podataka
Izvor: Autor (osobni primjer)

Važno je napomenuti kako ovo nisu svi stupci iz tablice nego su filtrirani za potrebe primjera. Isto tako, podaci su djelomično skriveni zbog zaštite privatnih podataka.

5.3. YouTube analiza

Za svaku aplikaciju postoji više setova podataka. Neki setovi su pohranjeni kao jednostavne datoteke, a drugi su pohranjeni u jednu ili više SQLite baza podataka. U nekim slučajevima se želi usporedno prikazati dva ili više seta podataka. SQLite GUI alati mogu to prikazati, ali često je to nepregledno u obliku više tablica.

U sljedećem primjeru su uzeta dva seta podataka vezana uz *YouTube* aplikaciju. To su tablice *search_history* i *watch_history* iz *history.db* SQLite baze podataka. Navedene tablice sadržavaju podatke o povijesti pretraga i pregleda video zapisa. Logičan je zahtjev za usporednim pregledom. Kao što je navedeno, za ovaj slučaj u SQLite alatu postoji problem nepreglednosti. Ideja je izvesti tablice u HTML datoteke koje se mogu otvoriti *web* preglednikom i lakše usporediti. Kako to izgleda prikazano je na slici 15.

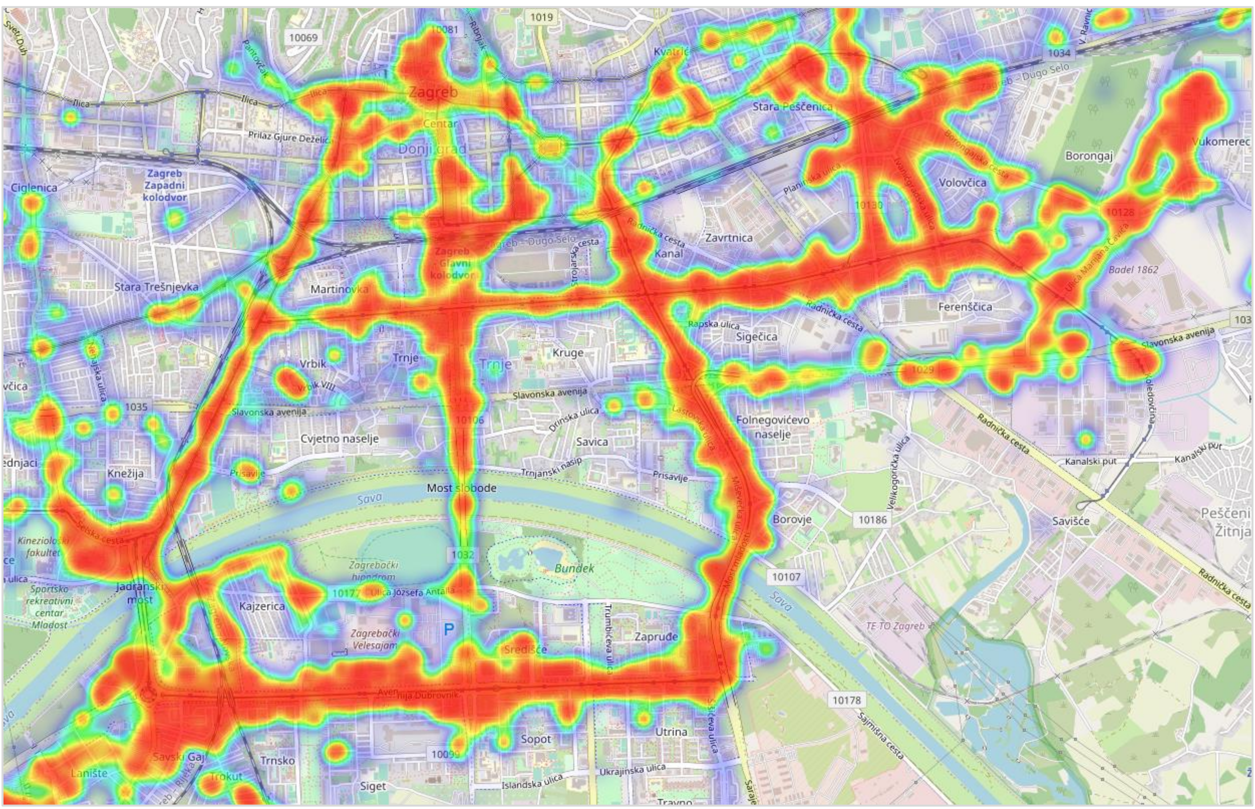


Slika 15. HTML prikaz podataka iz dvije tablice
Izvor: Autor (osobni primjer)

5.4. Google Maps analiza

Jedan od spomenutih formata datoteke je JSON (*engl. JavaScript Object Notation*). Analiza podataka *Google Maps* aplikacije je idealna za izvoz datoteke s ekstenzijom *.json*. Kod ove aplikacije također postoji *history.db* SQLite baza podataka. Najvažniji podaci su pohranjeni u tablici *location_history*.

Pregledom tablice u SQLite alatu se brzo shvaća zašto to nije najbolji način analize. Naime, pohranjeni podaci se uglavnom odnose na koordinate i vremenske oznake. Prvi logičan korak je izvesti tablicu u HTML obliku. Time se u ovom slučaju ne mijenja ništa. Dobije se malo drugačiji ispis, ali se koordinate i dalje moraju nekako interpretirati. Za ovako specifične slučajeve su razvijeni posebni vizualizacijski alati što je pokazano primjerom na slici 16.



Slika 16. Vizualni alat – prikaz *location_history* tablice
Izvor: [27], Autor (osobni primjer)

Prikaz karte učestalosti kretanja generiran je alatom *Location History Visualizer*, [27]. Potrebno je učitati JSON datoteku s podacima koja sadrži parove geografskih širina i dužina. Alat slijedno čita podatke i automatski generira ovakav prikaz iz kojeg se mnogo lakše mogu analizirati podaci o kretanju korisnika.

6. Oporavak izbrisanih podataka i mogućnost primjene

U prethodnom poglavlju je provedena ekstrakcija i analiza podataka. Radi se o logičkoj ekstrakciji koja se provodi putem većine softverskih forenzičkih alata. Pokazano je kako su dobiveni podaci sasvim dovoljni za osnovnu analizu i dobivanje informacija o korisničkim navikama. Ipak, navedenim postupkom nisu dobiveni svi mogući podaci.

Izbrisani podaci i dalje postoje u memorijskom prostoru, ali nisu indeksirani i zato ih postupak logičke ekstrakcije ne dohvaća. Ti podaci postoje sve do trenutka kada su prepisani novim podacima. Kako bi se pristupilo tim podacima potrebno je koristiti forenzički alat koji ima svojstva fizičke ekstrakcije. Oporavak izbrisanih podataka čini presudan aspekt forenzičke analize kada se ona izvodi u sklopu kaznenog ili prekršajnog postupka. Prilikom jednostavnijih edukacijskih i pokaznih primjera (kakav je proveden u prošlom poglavlju), oporavak ne čini nužno područje interesa. Ovo poglavlje pokriva primjer jednog forenzičkog alata s mogućnošću oporavka izbrisanih podataka te daje kratki pregled mogućih primjena.

6.1. Autopsy alat

Nakon provedene ekstrakcije u prošlom poglavlju je dobiven rezultat u obliku strukture direktorija u kojima se nalaze SQLite baze podataka te krajnji podaci. Izvođenjem procesa fizičke ekstrakcije dobiva se jedna datoteka. Vrsta datoteke je izvornog naziva *image file* što se u slobodnom prijevodu može interpretirati kao slika sustava. U literaturi se definira još kao forenzička slika ili sirova slika (*engl. Raw image*), [20].

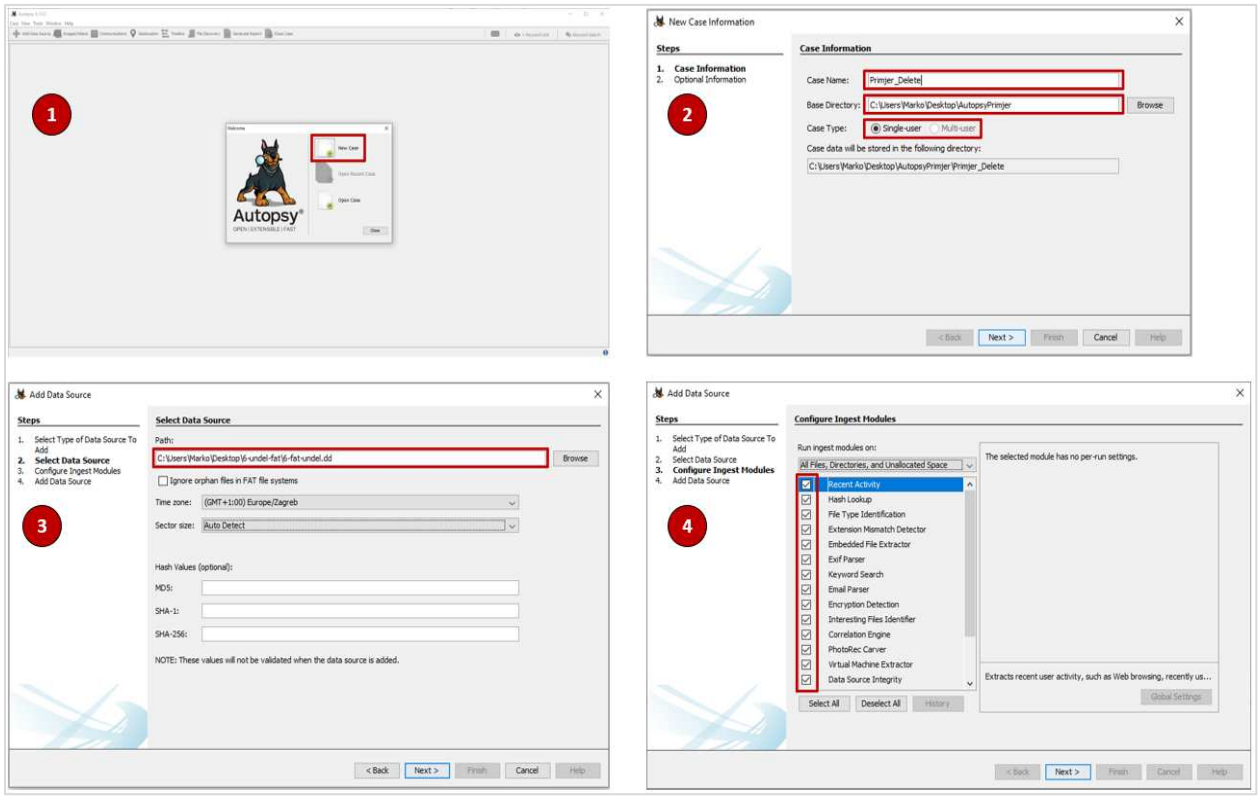
Razlika u odnosu na logičku ekstrakciju je nemogućnost definiranja ekstrakcije za određenu aplikaciju nego se predefinirano radi na razini sustava. To je dobro zato što je ionako potrebno obuhvatiti cijeli memorijski prostor. Pregled datoteke nije moguć putem datotečnog sustava nego je potrebno učitati sliku sustava u za to definirani alat.

U slučaju *Android* platforme postoji mnogo alata za fizičku ekstrakciju slike sustava, a najpoznatiji su *Cellebrite* i XRY alati, [20]. Alati s opcijom fizičke ekstrakcije mnogo češće dolaze s komercijalnom licencom koja se plaća, dok je slučaj kod alata za logičku ekstrakciju pokazivao više alata sa slobodnom licencom. Za analizu slike sustava najboljim se pokazao *Autopsy* forenzički alat. Ima slobodnu licencu, a izvršna instalacijska datoteka je dostupna za preuzimanje na službenoj stranici, [28]. *Autopsy* je prvenstveno razvijen za čitanje slike sustava nastale uporabom forenzičkih alata iz *Sleuth Kit* kolekcije, ali danas ima podršku za učitavanje slike sustava nastale iz većine dostupnih forenzičkih alata.

6.1.1. Učitavanje slike sustava u Autopsy alat

Nakon preuzimanja i instalacije *Autopsy* alata potrebno je učitati sliku sustava. Postupak se obavlja u četiri koraka, [20] koji su prikazani na slici 17:

1. Odabir opcije “Stvori novi slučaj.”
2. Unos svih potrebnih podataka o slučaju (naziv slučaja, lokaciju za pohranu slučaja i tip slučaja).
3. Definicija putanje do lokacije gdje je pohranjena slika sustava.
4. Konfiguracija modula za pregled koji će se pokrenuti za zadanu sliku sustava.

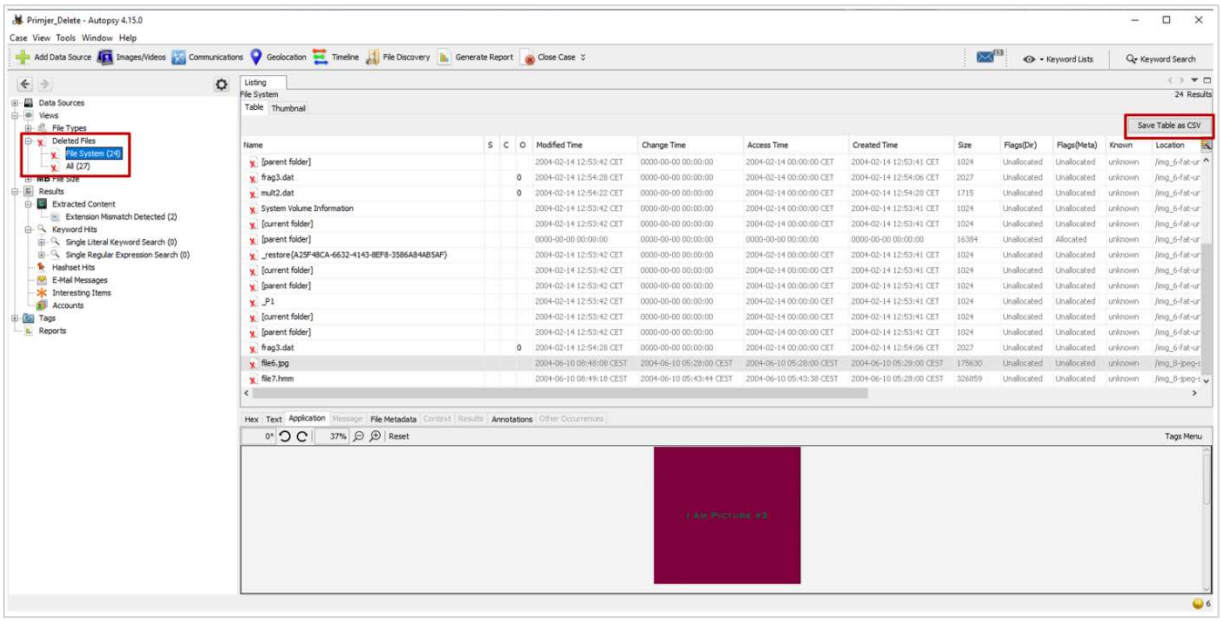


Slika 17. Proces učitavanja slike sustava u *Autopsy*
Izvor: Autor (osobni primjer)

Autopsy sučelje je intuitivno te ne zahtijeva posebna znanja iz područja fizičke ekstrakcije podataka. Nakon odabira opcije završetka počinje uvoz i interpretacija slike sustava. Taj proces traje osjetno duže nego kod logičke ekstrakcije, a razlog tome je veličina datoteke slike sustava koja danas iznosi više desetaka GB za gotovo sve *Android* uređaje.

6.1.2. Analiza slike sustava u Autopsy alatu

Nakon završetka čitanja slike sustava otvara se dio sučelja za analizu dobivenih podataka. Glavni direktorij ovdje je *Data Sources*, a sa slike 18 je vidljivo postojanje pogleda koji olakšavaju analizu podataka. Posebno zanimljiv je pogled na izbrisane podatke. Sa slike je vidljivo kako je uspješno prikazana izbrisana slikovna datoteka. U slučaju da je određena datoteka dijelom prepisana novim podacima i dalje se može vidjeti dio originalnog zapisa u binarnom obliku. Poveznicu sa SQLite bazom podataka čini označeni gumb kojim se dobiveni rezultati izvoze u tabličnom CSV obliku. Time je moguće pristupiti izbrisanim datotekama i putem SQLite alata.



Slika 18. Primjer izgleda slike sustava otvorene u *Autopsy* alatu
Izvor: [29], Autor (osobni primjer)

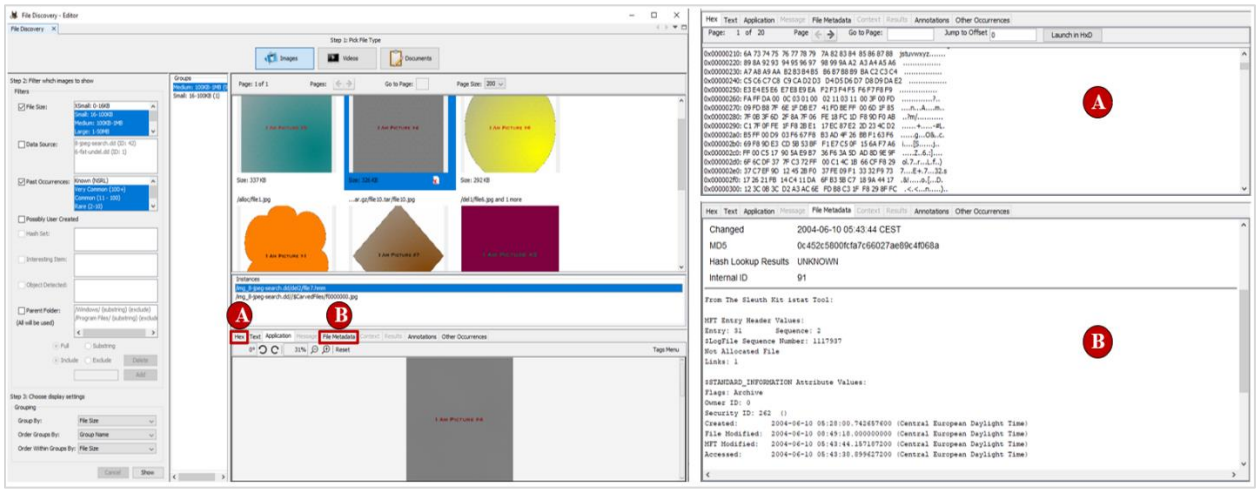
Još jedna razlika u odnosu na logičku ekstrakciju je pristup direktorijima. Ovdje se i dalje radi o datoteci slike sustava pa je zato pristup direktorijima moguć jedino preko *Autopsy* ili sličnog alata, a ne preko računalnog datotečnog sustava osim ako se podaci ne izvezu kako je gore navedeno.

Izbrisane datoteke postoje u memoriji u cijelosti ili djelomično, samo nisu više indeksirane. Najčešće su to slikovne datoteke. *Autopsy* analizira zaglavlje izbrisane datoteke iz čega saznaje duljinu zapisa u memoriji što omogućuje oporavak.

Dio memorije koji sadrži određene zapise koji se ne mogu automatski oporaviti se nalazi u direktoriju *$Unalloc*. U tom slučaju ispitivač može pomoću *Autopsy* alata ručno čitati binarne zapise u tom djelu memorije kako bi izvukao korisne informacije, [20].

6.2. Analiza oporavljenih podataka

Otvaranjem dobivenih podataka u SQLite grafičkom alatu dobije se vrlo sličan prikaz. U ovom slučaju forenzički istražitelji nemaju potrebu za time. Naime, *Autopsy* nudi napredne funkcije za analizu podataka. Najkorištenija funkcija je pregled putem opcije *File Discovery* koja je dostupna u alatnoj traci. Odabirom se pokreće prikaz kao na slici 19.



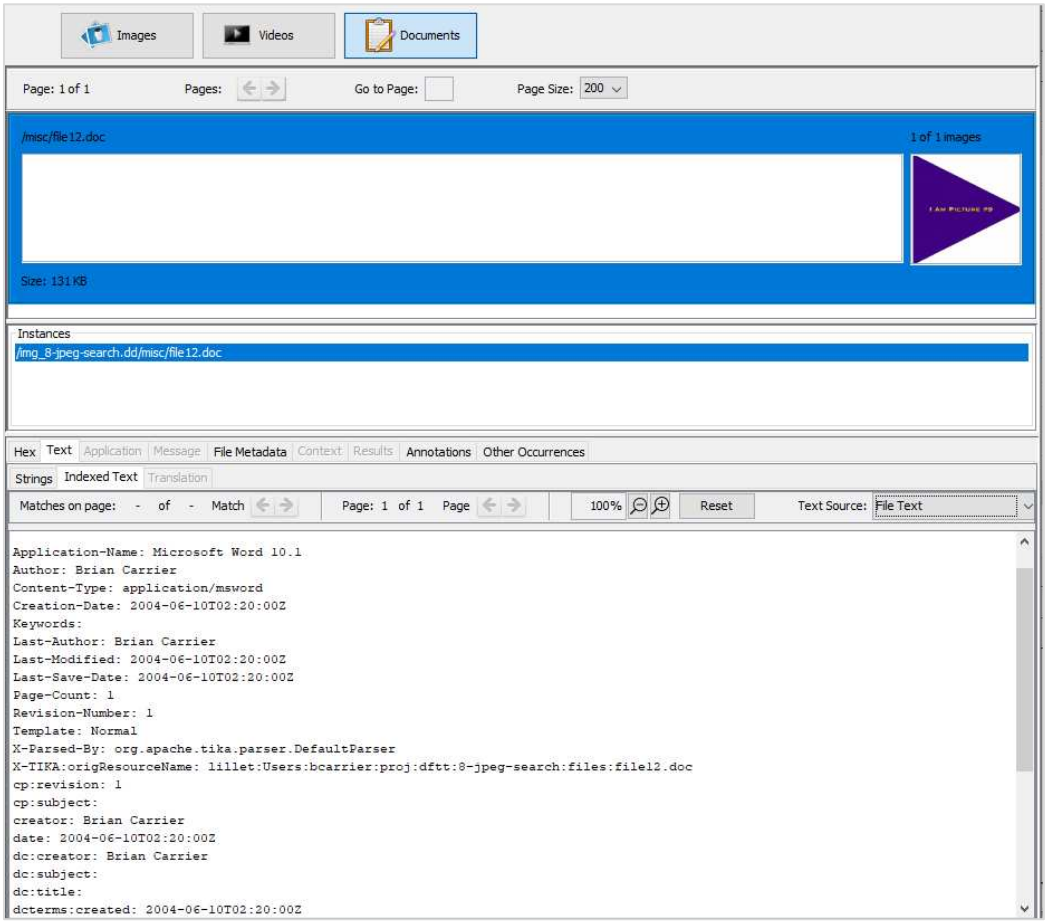
Slika 19. Prikaz analize slikovnih datoteka u *Autopsy* alatu
Izvor: [29], Autor (osobni primjer)

U otvorenom prozoru je dostupan pregled svih podataka ovisno o tipu datoteke. Na slici 19 je dan prikaz slikovnih datoteka koje su uspješno oporavljene nakon što su bile izbrisane. *Autopsy* alat ima mogućnost rekonstrukcije kao što se vidi iz slike 19. SQLite alatom su dostupne neke informacije o datotekama kao što su mjesto gdje je datoteka bila pohranjena ili *Hash* vrijednost datoteke što nije dovoljno za rekonstrukciju.

Osim rekonstrukcije dostupne su i druge informacije o slikovnoj datoteci. U gornjem desnom kutu slike 19 označenom slovom A prikazan je izgled memorije gdje je slika zapisana. Konkretno, radi se o standardnom heksadecimalnom zapisu. Takav zapis je vrlo koristan u slučajevima kad je slika djelomično prepisana novom datotekom te ju alat ne može u potpunosti rekonstruirati. Tada stručnjaci mogu dobiti određene informacije iz heksadecimalnog zapisa.

Dio slike 19 označen slovom B prikazuje metapodatke odabrane slikovne datoteke. Takvi podaci otkrivaju sve dodatne informacije koje su vezane uz datoteku. Tu je moguće otkriti vlasnika ili kreatora datoteke. Također, zapisane su sve vremenske oznake poput vremena kreiranja, zadnjeg pristupa, modificiranja te brisanja datoteke. Osim toga, metapodaci čuvaju informacije o početnom mjestu u memoriji te o veličini datoteke.

Na slici 20 je dan prikaz tekstualne datoteke koja je također bila izbrisana, ali je oporavljena iz nealociranog memorijskog prostora.



Slika 20. Prikaz analize tekstualne datoteke u *Autopsy* alatu
Izvor: [29], Autor (osobni primjer)

Za razliku od slikovnih datoteka, ovdje najviše informacija daje odabir opcije *Text*. Kod slikovnih datoteka ta opcija prikazuje nasumičan niz znakova ovisno o načinu pohrane heksadecimalnog zapisa u memoriji.

U slučaju prikazane tekstualne datoteke dostupna je informacija o softveru u kojem je kreirana datoteka. Ovdje se radi o *Microsoft Word* verziji 10.1. Tu je dostupan i velik broj informacija koje su vidljive u metapodacima. Ipak, ovakav tekstualni prikaz se može koristiti za prikaz naslova i sadržaja datoteke ako ona nije prazna.

Sadržaj se može čitati jednostavnim otvaranjem datoteke nakon što je ona oporavljena, ali otvaranjem datoteke se automatski mijenjaju metapodaci čime se narušava pravilo forenzičke analize. Zato je važno analizirati datoteke unutar slučaja otvorenog u alatu.

6.3. Mogućnost primjene oporavka podataka

Oporavak izbrisanih podataka je najznačajnija mogućnost forenzičke analize. Sposobnost povratka izbrisanih datoteka je danas presudna za rješavanje većine kaznenih slučajeva. Sa stajališta prosječnog korisnika vraćanje podataka obično se odnosi na ugrađena rješenja operativnog sustava poput strukture koša za smeće (*engl. Recycle Bin*). Ipak, sve veća razina edukacije korisnika omogućuje nova znanja koja pomažu pri željenom skrivanju podataka.

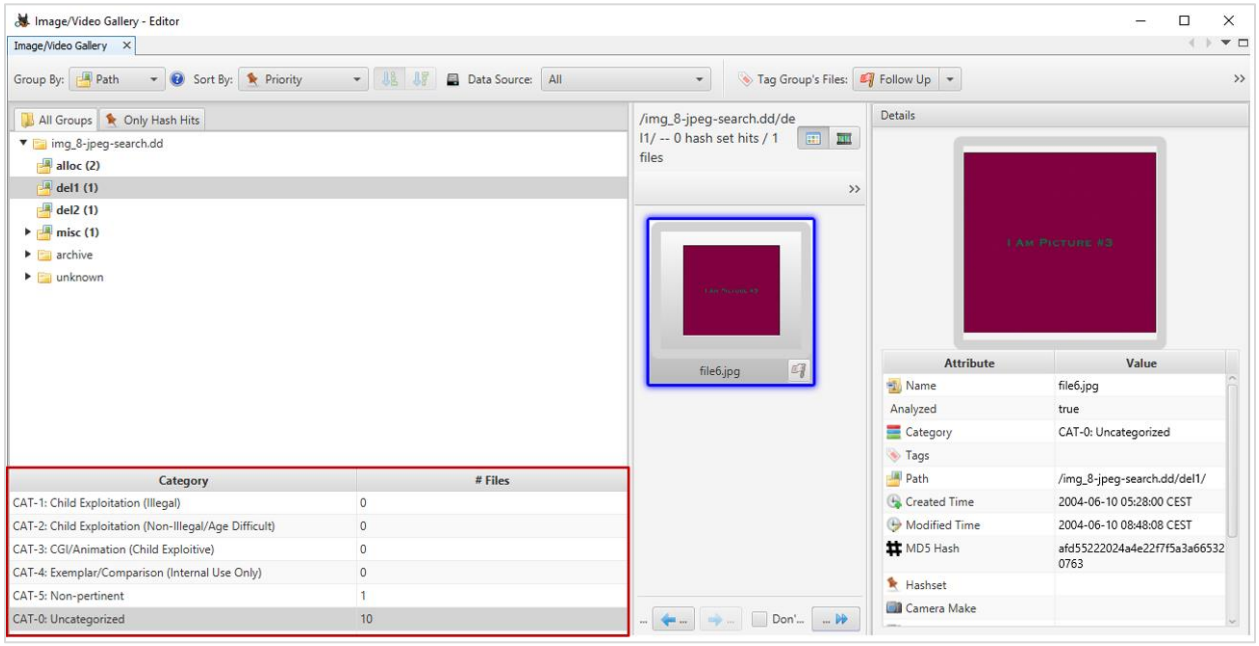
Unatoč tome, s *Android* platforme moguće je vratiti većinu izbrisanih podataka ako je uređaj ispravno prikupljen. Međutim, ako se ne postupa pravilno tijekom rukovanja s uređajem, izbrisani podaci mogu biti zauvijek izgubljeni. Kako bi se osiguralo da izbrisani podaci ne budu prepisani, preporučuje se zabrana korištenja uređaja za bilo koju aktivnost nakon njegove akvizicije. Bilo koja radnja može utjecati na dodavanje novih podataka koji mogu biti zapisani na mjesto gdje se nalaze izbrisane datoteke. Dakle, do početka provedbe procesa forenzičke ekstrakcije, uređaj bi idealno trebao biti potpuno izoliran.

Svi *Android* datotečni sustavi imaju metapodatke koji sadrže informacije o datotečnoj hijerarhiji, nazivima datoteka i statusu aktivnosti. Brisanjem se neće nepovratno ukloniti podaci i što je još važnije metapodaci. Čitanjem zaglavlja datoteke pronalaze se metapodaci, a upravo na taj način radi *Autopsy*, [20]. Za sve što nije uspjelo biti pročitano postoji opcija ručne analize ispitivača.

Kada se govori o mogućnostima oporavka podataka s pametnih mobilnih telefona, problematika se dijeli na dva područja. Jednostavniji pristup je u slučaju **vraćanja izbrisanih podataka s vanjske pohrane** (najčešće SD memorijska kartica). Podaci prisutni na SD karticama mogu otkriti mnoge informacije forenzičkim istražiteljima. Na vanjskoj pohrani će se najčešće oporaviti slikovne datoteke, videozapisi i glasovne snimke. SD kartica se može povezati s forenzičkim alatom kao vanjski uređaj za masovno pohranjivanje. Koristi se datotečni sustav FAT32 (*engl. File Allocation Table*) koji je podržan u većini operativnih sustava i zato je pristup oporavku podataka vanjske pohrane jednostavan, [20].

Oporavak izbrisanih podataka iz interne memorije *Android* uređaja predstavlja nešto kompliciraniju problematiku. Ovdje se u većini slučajeva radi o podacima aplikacija, porukama, kontaktima i pohranjenim računima. U idealnom slučaju će alati poput *Autopsy-a* generirati prikaz kao na prethodnoj slici bez problema, ali često zna doći i do pogreške u pristupu internoj memoriji. Jedan od najvećih problema je nemogućnost alata u pristupu uređaja ako on nije u potpunosti sistemski otključan (*engl. rooted*). Otključavanjem uređaja može doći do izmjene podataka u internoj memoriji. Nadalje, koriste se datotečni sustavi specifični za *Android* platformu. Kada se uz sve navedeno u obzir uzme i problematika pristupa SQLite bazama podataka, jasno je u kojoj mjeri je mogućnost oporavka podataka interne memorije teža od vanjske pohrane, [20].

Unutar postupka forenzičke analize čest je slučaj kategorizacija podataka. *Autopsy* nudi tu mogućnost u sklopu otvorenog slučaja. Primjer pregleda i kategorizacije se nalazi na slici 21.



Slika 21. Mogućnost kategorizacije u sklopu analize podataka
Izvor: [29], Autor (osobni primjer)

Forenzička analiza često se provodi u sklopu prekršajnih i kaznenih postupaka. Rezultati moraju biti prikazani u razumljivom obliku. Podjela po kategorijama je jednostavan način odvajanja podataka od interesa te svih ostalih podataka.

Na slici 21 se vidi kako se sve oporavljenje slike nalaze u kategoriji nula što zapravo znači da nisu kategorizirane. Ovim korakom analitičari i ostali stručnjaci pregledavaju slikovne datoteke te ih ovisno o potrebi smještaju u definirane kategorije, a postoji i mogućnost kreiranja novih kategorija. Crvenim okvirom na slici su prikazane predefinirane kategorije. Radi se većinom o ilegalnim sadržajima koji čine neosporive dokaze.

Nakon kategorizacije prilikom kreiranja izvještaja, utvrđuje se postojanje kategorija koje su potrebne za prikaz u sudskom postupku. Sličan postupak kategorizacije se provodi u ostalim tipovima datoteka. Upravo takvo izvješće čini najčešće korištenu mogućnost primjene za oporavljene podatke.

7. Zaključak

Pametni mobilni uređaji su u mnogo kategorija usporedivi s osobnim računalima, ali i ostalim terminalnim uređajima. Konkretno, razlike u arhitekturi, funkcionalnostima i procesorskoj moći su sve manje. Također, prostor za pohranu u današnjim uređajima je i više nego dovoljan za korisničke potrebe. Na prvi pogled, iz perspektive kompatibilnosti ne postoji prepreka za korištenje jedne od računalnih baza podataka (npr. *MySQL, Microsoft* SQL) na *Android* mobilnom uređaju. Ipak, detaljnijim pogledom na komunikacijsku arhitekturu SQLite baze podataka se zaključuje kako su njene prednosti u odnosu na ostale baze prevelike u slučaju pametnih mobilnih uređaja. Osim toga, ostale SQLite karakteristike samo potvrđuju zaključak kako je ovaj DBMS pravi odabir za uporabu na *Android* sustavu.

Daljnjom analizom strukture SQLite-a je pokazano da je integritet baze podataka osiguran. Instalacija i pokretanje SQLite alata su jednostavni bilo da se radi o naredbenom retku ili grafičkom sučelju. Odabir ovisi o korisniku. Jedna od osnovnih funkcija alata je kreiranje baze podataka i manipulacija nad podacima. Kada se u obzir uzme postupak forenzičke ekstrakcije, dobit će se set podataka za analizu. To znači da je SQLite alat u kontekstu ovog rada korišten u svrhu analize dostupnih baza podataka. Upravo to čini poveznicu dvije glavne odrednice diplomskog rada (SQLite baza podataka i proces forenzičke analize).

Odabir forenzičkog alata ne predstavlja problem iz razloga postojanja sličnog skupa funkcionalnosti kod većine alata. Za potrebe diplomskog rada je preporukom odabran *Andriller*. Navedeni alat obavlja ekstrakciju podataka na logičkoj razini. Ostale funkcionalnosti uključuju parsere za interpretaciju podataka i brisanje obrasca zaključavanja zaslona. Već je navedeno kako je analiza podataka provedena pomoću *DB Browser for SQLite* alata. To znači da je jedina potrebna funkcionalnost forenzičkog alata u ovom slučaju bila ekstrakcija i raspakiravanje podataka. Tu funkcionalnost posjeduju svi alati. Ipak, *Andriller* se pokazao kao vrlo intuitivan alat i rezultati su bili zadovoljavajući.

Kada se govori o oporavku izbrisanih podataka, postavlja se potreba za ekstrakcijom na fizičkoj razini. Time se dobiva slika cijelog sustava u jednoj datoteci. Toj datoteci se ne može pristupiti preko datotečnog sustava kao u slučaju logičke ekstrakcije. *Autopsy* alat je odabran za pregled slike sustava. Pregled slike sustava se uspješno izvršava, te je moguć pristup pojedinim izbrisanim datotekama. Preduvjet za fizičku ekstrakciju podataka putem određenog alata je sistemski otključan uređaj. Preporuka je prepustiti taj dio stručnjacima te koristiti datoteku slike sustava iz dostupnih studija slučaja.

Naime, prilikom nestručnog pristupa sistemskom otključavanju uređaja i fizičkoj ekstrakciji podataka vrlo lako može doći do neželjenog gubitka podataka, a u nekim slučajevima i do kvara uređaja.

Literatura

[1] MariaDB: Database Theory; Dostupno na: https://mariadb.com/kb/en/database-theory/. (pristupljeno: kolovoz 2020.)

[2] P. Sanderson; SQLite Forensics; Paul Sanderson, Porthleven, UK, 2018.

[3] J. A. Kreibich; Using SQLite; O'Reilly Media, Inc., Sebastopol, CA, 2010.

[4] D. R. Hipp; Home Page for D. Richard Hipp; Dostupno na: http://www.hwaci.com/drh/. (pristupljeno: kolovoz 2020.)

[5] Statista: Number of smartphone users worldwide from 2016 to 2021; Dostupno na: https://www.statista.com/statistics/330695/number-of-smartphone-users-worldwide/. (pristupljeno: kolovoz 2020.)

[6] J. Feiler; Introducing SQLite for Mobile Developers; Apress Media LLC, New York, NY, 2015.

[7] G. Da Rocha; Learning SQLite for iOS. Birmingham; Packt Publishing Ltd., UK; 2016.

[8] S. Kumar Aditya, V. Kumar Karn; Android SQLite Essentials; Packt Publishing Ltd., Birmingham, UK, 2014.

[9] S. T. Bhosale, T. Patil, P. Patil; SQLite: Light Database System; Int. J. Comput. Sci. Mob. Comput., vol. 4, no. 4, pp. 882–885, 2015.

[10] G. Allen, M. Owens; SQLite: the definitive guide; Apress Media LLC, New York, NY, 2010.

[11] Technopedia: Data Integrity; Dostupno na: https://www.techopedia.com/definition/811/data-integrity-databases. (pristupljeno: kolovoz 2020.)

[12] SQLite documentation: PRAGMA Statements; Dostupno na: https://www.sqlite.org/pragma.html. (pristupljeno: kolovoz 2020.)

[13] SQlite Pocket Reference; Dostupno na: https://digital-forensics.sans.org/media/SQlite-PocketReference-final.pdf. (pristupljeno: kolovoz 2020.)

[14] SQLite documentation: Write-Ahead Logging; Dostupno na: https://www.sqlite.org/wal.html. (pristupljeno: kolovoz 2020.)

[15] SQLite Download Page; Dostupno na: https://www.sqlite.org/download.html. (pristupljeno: kolovoz 2020.)

[16] SQLite Tutorial; Dostupno na: https://www.sqlitetutorial.net/. (pristupljeno: kolovoz 2020.)

[17] GitHub: SQLiteStudio; Dostupno na: https://github.com/pawelsalawa/sqlitestudio/releases. (pristupljeno: kolovoz 2020.)

[18] DBeaver Community; Dostupno na: https://dbeaver.io/. (pristupljeno: kolovoz 2020.)

[19] DB Browser for SQLite; Dostupno na: https://sqlitebrowser.org/. (pristupljeno: kolovoz 2020.)

[20] R. Tamma, O. Skulkin, H. Mahalik, S. Bommisetty; Practical mobile forensics; Packt Publishing Ltd., Birmingham, UK, 2018.

[21] S. Husnjak; Autorizirana predavanja, kolegij Forenzička analiza informacijsko komunikacijskog sustava - Metodologije forenzičke analize informacijsko komunikacijskog sustava (2019/20); Sveučilište u Zagrebu, Fakultet prometnih znanosti, Zagreb, 2020.

[22] L. Reiber; Mobile Forensic Investigations; McGraw-Hill Education, New York, NY, 2019.

[23] S. Husnjak; Autorizirana predavanja, kolegij Forenzička analiza informacijsko komunikacijskog sustava - Digitalni dokazi i ekstrakcija podataka mobilnih uređaja (2019/20); Sveučilište u Zagrebu, Fakultet prometnih znanosti, Zagreb, 2020.

[24] Github: Andriller CE; Dostupno na: https://github.com/den4uk/andriller. (pristupljeno: kolovoz 2020.)

[25] A. Gupta; Learning Pentesting for Android Devices; Packt Publishing Ltd., Birmingham, UK, 2014.

[26] J. Clement; Statista: WhatsApp - Statistics & Facts 2019; Dostupno na: https://www.statista.com/topics/2018/whatsapp/. (pristupljeno: kolovoz 2020.)

[27] Location History Visualizer; Dostupno na: https://locationhistoryvisualizer.com/heatmap/. (pristupljeno: kolovoz 2020.)

[28] Autopsy; Dostupno na: http://www.sleuthkit.org/autopsy/. (pristupljeno: kolovoz 2020.)

[29] B. Carrier; Digital Forensics Tool Testing Images; Dostupno na: http://dftt.sourceforge.net/. (pristupljeno: kolovoz 2020.)

Popis kratica

ACID	(Atomicity, Consistency, Isolation, Durability) svojstva transakcijske baze podataka
API	(Application Programming Interface) softverski posrednik za komunikaciju s aplikacijom
CPU	(Central Processing Unit) elektronička komponenta koja izvršava zadane instrukcije
CSV	(Comma Separated Value) vrsta datoteke za pohranu podataka
DBMS	(Database Management System) sustav upravljanja bazom podataka
FAT32	(File Allocation Table) datotečni sustav
GUI	(Graphical User Interface) grafičko sučelje za pristup sustavu ili uređaju
HTML	(Hyper Text Markup Language) standardni programski jezik za dokumente dizajnirane za prikaz u web pregledniku
JPG	(Joint Photographic Experts Group) format slikovnih datoteka
JSON	(JavaScript Object Notation) lagan i čitljiv format za strukturiranje i prijenos podataka
RAM	(Random Access Memory) razina memorije za kratkotrajnu pohranu
SHM	(Shared Memory File) pomoćna datoteka za indeksiranje WAL dnevnika
SMS	(Short Message Service) usluga razmjene tekstualnih poruka
SQL	(Structured Query Language) strukturirani jezik za definiciju baze podataka i upite nad bazom podataka
STB	(Set Top Box) uređaj za pretvorbu izvornog signala u sadržaj za prikaz na zaslonu
USB	(Universal Serial Bus) sučelje za povezivanje perifernih uređaja s računalom
VDBE	(Virtual Database Engine) virtualni stroj – centralni dio SQLite arhitekture
WAL	(Write Ahead Log) vrsta dnevnika za očuvanje integriteta SQLite baze podataka

Popis slika

Popis grafičkih prikaza

Popis tablica

Popis tekstualnih priloga

Sveučilište u Zagrebu
Fakultet prometnih znanosti
10000 Zagreb
Vukelićeva 4

IZJAVA O AKADEMSKOJ ČESTITOSTI I SUGLASNOST

Izjavljujem i svojim potpisom potvrđujem kako je ovaj diplomski rad isključivo rezultat mog vlastitog rada koji se temelji na mojim istraživanjima i oslanja se na objavljenu literaturu što pokazuju korištene bilješke i bibliografija.

Izjavljujem kako nijedan dio rada nije napisan na nedozvoljen način, niti je prepisan iz necitiranog rada, te nijedan dio rada ne krši bilo čija autorska prava.

Izjavljujem također, kako nijedan dio rada nije iskorišten za bilo koji drugi rad u bilo kojoj drugoj visokoškolskoj, znanstvenoj ili obrazovnoj ustanovi.

Svojim potpisom potvrđujem i dajem suglasnost za javnu objavu diplomskog rada pod naslovom **Forenzička analiza SQLite baze podataka Android uređaja**

na internetskim stranicama i repozitoriju Fakulteta prometnih znanosti, Digitalnom akademskom repozitoriju (DAR) pri Nacionalnoj i sveučilišnoj knjižnici u Zagrebu.

U Zagrebu, 10.9.2020

Student/ica:

(potpis)