دانشگاه صنعتی امیرکبیر

(پلی تکنیک تهران)

دانشکده مهندسی کامپیوتر و فن‌آوری اطلاعات

پایان‌نامه کارشناسی

گرایش نرم‌افزار

عنوان

طراحی و پیاده‌سازی ابزاری با قابلیت توسعه‌پذیری به منظور استخراج داده‌های پاک‌شده از گوشی‌های هوشمند

نگارش

احسان عدالت

استاد راهنما

دکتر بابک صادقیان

شهریور ۱۳۹۴

به نام خدا

تعهدنامه اصالت اثر

تاریخ:

دانشگاه صنعتی امیرکبیر
(پلی تکنیک تهران)

اینجانب احسان عدالت متعهد می‌شوم که مطالب مندرج در این پایان نامه حاصل کار پژوهشی اینجانب تحت نظارت و راهنمایی اساتید دانشگاه صنعتی امیرکبیر بوده و به دستاوردهای دیگران که در این پژوهش از آنها استفاده شده است مطابق مقررات و روال متعارف ارجاع و در فهرست منابع و مآخذ ذکر گردیده است. این پایان نامه قبلاً برای احراز هیچ مدرک هم‌سطح یا بالاتر ارائه نگردیده است.

در صورت اثبات تخلف در هر زمان، مدرک تحصیلی صادر شده توسط دانشگاه از درجه اعتبار ساقط بوده و دانشگاه حق پیگیری قانونی خواهد داشت.

کلیه نتایج و حقوق حاصل از این پایان نامه متعلق به دانشگاه صنعتی امیرکبیر می‌باشد. هرگونه استفاده از نتایج علمی و عملی، واگذاری اطلاعات به دیگران یا چاپ و تکثیر، نسخه‌برداری، ترجمه و اقتباس از این پایان نامه بدون موافقت کتبی دانشگاه صنعتی امیرکبیر ممنوع است. نقل مطالب با ذکر مآخذ بلامانع است.

احسان عدالت

امضا

با سپاس از سه وجود مقدس:

آنان که ناتوان شدند تا ما به توانایی برسیم...

موهایشان سپید شد تا ما روسفید شویم...

و عاشقانه سوختند تا گرمابخش وجود ما و روشنگر راهمان باشند...

پدرانمان

مادرانمان

استادانمان

تقدیر و تشکر:

سپاس و ستایش مر خدای را جل و جلاله که آثار قدرت او بر چهره روز روشن، تابان است و انوار حکمت او در دل شب تار، درفشان. آفریدگاری که خویشتن را به ما شناساند و درهای علم را بر ما گشود و عمری و فرصتی عطا فرمود تا بدان، بنده ضعیف خویش را در طریق علم و معرفت بیازماید.

بدون شک جایگاه و منزلت معلم، بالاتر از آن است که در مقام قدردانی از زحمات بی شائبه‌ی او، با زبان قاصر و دست ناتوان، چیزی بنگارم. اما از آنجا که تجلیل از معلم، سپاس از انسانی است که هدف آفرینش را تامین می‌کند، به رسم ادب دست به قلم برده‌ام، باشد که این خردترین بخشی از زحمات آنان را سپاس گوید.

از پدر و مادر مهربانم، این دو معلم بزرگوار که همواره بر کوتاهی من، قلم عفو کشیده و کریمانه از کنار غفلت‌های گذشته‌اند و در تمام عرصه‌های زندگی یار و یاورم بوده‌اند؛

از استاد با کمالات، جناب آقای دکتر بابک صادقیان که در کمال سعه صدر، با حسن خلق و فروتنی، از هیچ کمکی در این عرصه بر من دریغ نداشتند؛

از اساتید محترم، جناب آقای دکتر مهران سلیمان فلاح و آقای مهندس بهمن پوروطن که زحمت داوری این رساله را متقبل شدند؛

و در پایان، از حمایت‌ها و دلسوزی‌های دوستان عزیزم، آقایان سید محمد مهدی احمدپناه، حمیدرضا رمضانی و احمد اسدی که در طول پروژه از راهنمایی‌هایشان استفاده کردم؛

کمال تشکر و قدردانی را دارم.

خلاصه پایان‌نامه

در این پروژه هدف، طراحی و پیاده‌سازی ابزاری با قابلیت توسعه‌پذیری به منظور استخراج داده‌های پاک‌شده از گوشی‌های هوشمند است. امروزه گوشی‌های هوشمند به دلیل استفاده آسان و گستره قیمتی متنوع در میان مردم به طور فراگیر گسترش پیدا کرده‌اند. این ابزارها مبتنی بر سیستم عامل هستند که نرم‌افزارهای گوناگونی بر روی این ابزارها قابل اجرا است.

از جمله این نرم‌افزارها می‌توان به نرم‌افزار پیام‌رسان، ایمیل، نرم‌افزار نگهداری عکس، نرم‌افزار یافتن موقعیت جغرافیایی و نرم‌افزار مرور اینترنت اشاره کرد. همان طور که مشخص است هر یک از این نرم‌افزارها اطلاعات زیادی از دارنده گوشی در خود ذخیره کرده‌اند.

امروزه از این اطلاعات برای انجام تحقیقات و اثبات جرم به وسیله پلیس و مراجع قضایی استفاده می‌شود. این موضوع اهمیت بازیابی داده‌های پاک‌شده را دوچندان می‌کند، چرا که ممکن است این داده‌ها عمدا از سیستم پاک شوند.

نرم‌افزارها داده‌ها را به شکل‌های گوناگونی در خود ذخیره می‌کنند که یکی از این روش‌ها استفاده از پایگاه‌داده Sqlite است. در این پروژه روش‌های استخراج و بازیابی اطلاعات، به روش منطقی از این پایگاه‌داده مورد بررسی قرار گرفته است.

ابزار پیاده‌سازی شده در این پروژه دارای این ویژگی است که پایگاه‌داده مورد استفاده نرم‌افزار انتخابی، از گوشی تلفن کپی شده و داده‌های آن به تفکیک جداول موجود در پایگاه‌داده، استخراج می‌شود. همچنین داده‌های پاک‌شده به سه روش، استخراج می شوند که عبارتند از بازیابی از فضای بلااستفاده، بازیابی از بلوک‌های آزاد و بازیابی به کمک فایل ژورنال.

همان طور که ذکر شد، گوشی‌های هوشمند، مبتنی بر سیستم عامل‌های گوناگونی هستند که برای هر کدام روشی خاص برای استخراج فایل پایگاه‌داده نیاز است. نرم‌افزار پیاده‌سازی شده دارای قابلیت توسعه‌پذیری است که به واسطه آن می‌توان روش‌های مختلف اتصال و دریافت فایل پایگاه‌داده از سیستم عامل‌ها و پلتفرم‌های مختلف را پیاده‌سازی و به سیستم اضافه کرد. در این پروژه، روش مورد استفاده برای دریافت فایل پایگاه‌داده از سیستم عامل اندروید ارائه شده‌است.

واژه‌های کلیدی:

پایگاه‌داده Sqlite، استخراج داده‌های پاک‌شده، توسعه‌پذیری، سیستم عامل اندروید

فهرست عناوین

صفحه

فهرست اشکال

صفحه

فهرست جداول

صفحه

۱

فصل اول

مقدمه

در پروژه حاضر هدف، طراحی و پیاده‌سازی ابزاری است که به کمک آن بتوان داده‌های ذخیره شده بر روی گوشی‌های تلفن هوشمند[۱] را استخراج کرد. ابزار مورد نظر به گونه‌ای پیاده‌سازی شده است که از قابلیت توسعه‌پذیری[۲] پشتیبانی می‌کند. در ادامه دلیل انتخاب گوشی‌های هوشمند، داده‌هایی که برای استخراج مورد نظر است و همچنین علت پیاده‌سازی قابلیت توسعه‌پذیری و ضرورت استفاده از آن، مورد بررسی قرار می‌گیرند. علاوه بر آن ابزارهای مشابه معرفی شده، تمایز و علل برتری ابزار حاضر بیان می‌شود.

۱.۱ گوشی‌های تلفن همراه هوشمند

امروزه با پیشرفت فناوری، ابزارهای دیجیتالی گوناگونی از قبیل رایانه‌های همراه، تلفن‌های همراه و تبلت‌ها به صورت گسترده در اختیار عموم مردم است. این ابزارها در سال‌های اخیر به دلیل قیمت مناسب و نیز فراگیری آسان، استفاده از آنها در میان مردم محبوبیت پیدا کرده است. در میان این ابزارها، گوشی‌های هوشمند در میان مردم جایگاه ویژه‌ای دارند و استفاده از آنها امری رایج و اجتناب ناپذیر است.

گوشی‌های هوشمند همانند رایانه‌ها دارای سیستم عامل هستند با این تفاوت که در سیستم عامل‌های استفاده شده در گوشی‌های تلفن همراه هوشمند به دلیل محدودیت منابع از جمله حافظه و باتری نحوه پیاده‌سازی نرم‌افزارها و مدیریت منابع متفاوت است. در ادامه برخی از پرکاربردترین

---

[۱] Smart Phones

[۲] Extensibility

نرم‌افزارها و کاربردهایی که عمدتا در اینگونه ابزارها به طور معمول وجود دارد مورد بررسی قرار می‌گیرند.

نرم‌افزار ارسال و دریافت پیام[۳]: گوشی‌های تلفن‌همراه هوشمند دارای نرم‌افزاری برای ارسال و دریافت پیام هستند که معمولا پیام‌های ارسال یا دریافت شده در آنها به تفکیک مخاطب، ذخیره شده است.

نرم‌افزار مخاطبان دارنده تلفن[۴]: این نرم افزار قادر است که اطلاعات افرادی که دارنده تلفن با آنها در ارتباط است را در خود ذخیره کند. از جمله این اطلاعات می‌توان به نام، شماره تلفن، آدرس و آدرس پست الکترونیک اشاره کرد.

نرم‌افزار ثبت تماس‌ها[۵]: تماس‌های اخیر دارنده تلفن در این نرم‌افزار ذخیره می‌شود. که معمولا تعدادی محدود از این اطلاعات (به طور مثال ۲۰ تماس اخیر) در اختیار دارنده گوشی قرار می‌گیرد.

نرم‌افزار مرورگر وب: از این نرم‌افزار برای دیدن صفحات وب استفاده می‌شود. علاوه بر سایت‌های بازدید شده و ذخیره شده، نام کاربری و گذرواژه ورود به برخی از سایت‌ها نیز توسط این نرم‌افزار نگهداری می‌شود.

نرم‌افزارهایی که در بالا بیان شدند، تعدادی از پرکاربردترین نرم‌افزارهایی هستند که در تمام گوشی‌های هوشمند با هر سیستم عاملی به طور پیش فرض وجود دارند. نرم افزارهای دیگری هم هستند که کاربر با توجه به نیاز خود و با توجه به سیستم عامل موجود در گوشی، اقدام به نصب و استفاده از آنها می‌کند که این نرم‌افزارها می‌توانند یک سری از اطلاعات را در خود ذخیره کنند. از جمله نرم‌افزارهای پرکاربرد که نسخه‌های مبتنی بر سیستم عامل‌های مختلف از آنها موجود است، می‌توان به وایبر[۶]، لاین[۷] و جی‌میل[۸] اشاره کرد.

---

[۳] Messaging

[۴] Contacts

[۵] Call Logs

[۶] Viber

همان‌طور که از بررسی نرم‌افزارهای بالا مشخص است، می‌توان گفت گوشی‌های هوشمند می‌توانند اطلاعات زیادی از دارنده خود را ذخیره کنند. این اطلاعات می‌تواند شامل اطلاعات افرادِ در رابطه با دارنده گوشی، مکالمات افراد، تماس‌های افراد باهم، سایت‌های بازدید شده و غیره باشند. امروزه به دلیل اهمیت اطلاعات موجود در گوشی‌های تلفن همراه هوشمند پلیس از آنها برای اثبات جرم و انجام تحقیقات استفاده می‌کند. پس علاوه بر اطلاعات جاری و موجود در گوشی، اطلاعات پاک‌شده نیز از اهمیت بالایی برخوردارند چرا که ممکن است این اطلاعات به شکل عمدی از سیستم پاک شده باشند. پس علاوه بر مکانیزمی برای استخراج اطلاعات باید مکانیزمی برای بازیابی اطلاعات از سیستم ارائه شود.

۲.۱ روش‌های استخراج اطلاعات از گوشی‌های هوشمند

روش‌های استخراج اطلاعات از سیستم به سه دسته کشف و استخراج دستی[۹]، منطقی[۱۰] و فیزیکی[۱۱] تقسیم‌بندی می‌شوند.

در روش دستی با استفاده از رابط کاربری تعبیه شده توسط نرم‌افزار و سیستم عامل، اطلاعات استخراج می‌شود. مشکل این روش در این است که تنها اطلاعات قابل مشاهده توسط نرم‌افزار و سیستم عامل قابل دسترسی است. پس اطلاعات پاک‌شده از طریق کاربر قابل مشاهده نیستند.

در روش منطقی، اطلاعات به صورت بایت به بایت از رسانه ذخیره‌سازی که در آن به شکل فایل و دایرکتوری ذخیره شده‌اند، استخراج می‌شود. این داده‌های خام باید پردازش شوند تا به واسطه آنها بتوان همه اطلاعات، چه پاک‌شده و چه پاک‌نشده را استخراج کرد. برای

---

[۷] Line

[۸] Gmail

[۹] Manual acquisition

[۱۰] Logical acquisition

[۱۱] Physical acquisition

استخراج اطلاعات پاک‌شده در این روش باید ساختار ذخیره‌سازی اطلاعات، مورد مطالعه قرار گیرد تا امکان استخراج اطلاعات پاک‌شده مورد سنجش قرار گیرد.

در روش فیزیکی، داده‌ها به شکل بیت به بیت از رسانه ذخیره‌سازی استخراج می‌شوند. در این روش با دسترسی مستقیم به حافظه فلش اطلاعات استخراج می‌شود.

۳.۱ نمونه‌ای از ابزارهای موجود

از جمله ابزارهای موجود در این حوزه می توان به Device seizure اشاره کرد. این نرم‌افزار دارای قابلیت استخراج اطلاعات از هر دو روش منطقی و فیزیکی است. علاوه بر آن، این نرم‌افزار از سیستم عامل‌های گوناگون از جمله اندروید[۱۲]، iOS و غیره پشتیبانی می‌کند. این نرم‌افزار قادر است اطلاعات مخاطبان، پیام‌ها، مکان‌های جغرافیایی که توسط GPS ثبت شده است، عکس‌ها و غیره را استخراج کند. همچنین می‌تواند اطلاعات پاک‌شده را نیز بازیابی کند.

با توجه به اطلاعاتی که در سایت این نرم‌افزار آمده است[۱] این طور به نظر می‌رسد که این نرم‌افزار جامع بوده و توانایی استخراج و بازیابی اطلاعات از گوشی‌های مختلف را دارد. اما مشکل عمده این ابزار این است که به صورت متن‌باز[۱۳] ارائه نشده است و با توجه به حساسیت بالای کاربرد این نرم‌افزار در حوزه‌های حقوقی و قضایی نمی‌توان به آن اعتماد کرد. همچنین نرم‌افزار متن‌بازی که به جامعیت Device seizure باشد وجود ندارد. بنابراین هدف از این پروژه پیاده‌سازی نرم‌افزاری مشابه Device seizure اما متن‌باز است که در ادامه ویژگی‌ها و محدوده‌های آن بیان می‌شود.

[۱۲] Android

[۱۳] Open Source

۴.۱ نمای کلی پروژه

در این پروژه از روش استخراج و بازیابی اطلاعات به شکل منطقی استفاده خواهد شد. در این روش هدف، استخراج و بازیابی اطلاعات با استفاده از ساختار فایل‌ها و سیستم مدیریت فایل[۱۴] موجود بر روی سیستم عامل است. فایل‌ها دارای قالب[۱۵]های گوناگون هستند که می‌توان به قالب‌های عکس و قالب فایل پایگاه‌داده‌ها اشاره کرد. برای بازیابی یا استخراج، نیاز است تا با ساختار سیستم مدیریت فایل یا ساختار فایل هدف آشنا شد تا پس از پردازش‌های لازم بتوان اطلاعات را استخراج یا بازیابی نمود.

در نرم‌افزارهای گوشی‌های هوشمند برای ذخیره‌سازی اطلاعات از پایگاه‌داده استفاده می‌شود. با توجه به محدودیت‌های استفاده از منابع در گوشی‌های هوشمند بهترین انتخاب پایگاه‌داده Sqlite است که متن‌باز بوده و نرم‌افزارهای مبتنی بر سیستم عامل‌های اندروید، iOS و بسیاری از نرم‌افزارهای مبتنی بر ویندوز از این پایگاه‌داده استفاده می‌کنند.(تمام نرم‌افزارهای ذکر شده در بالا از پایگاه‌داده Sqlite استفاده می‌کنند.) موتور این پایگاه‌داده به صورت پیش فرض بر روی این سیستم عامل‌ها قرار دارد تا توسعه‌دهندگان نرم‌افزارها بتوانند به راحتی از این پایگاه‌داده استفاده کنند.

در این پروژه، هدف پیاده‌سازی روشی برای استخراج و بازیابی اطلاعات از پایگاه‌داده Sqlite است. برای این کار نیاز است تا کتابخانه‌ای پیاده‌سازی شود تا پردازش فایل پایگاه‌داده را بر عهده بگیرد. علاوه بر آن نیاز است تا کتابخانه‌ای وظیفه برقراری ارتباط با گوشی هوشمند را بر عهده بگیرد تا به وسیله آن بتوان فایل‌های پایگاه‌داده را از گوشی به رایانه منتقل کرد.

از آنجایی که روش‌های برقراری ارتباط با گوشی‌های مختلف با سیستم عامل‌های مختلف متفاوت است، نیاز است تا مکانیزمی برای این منظور پیاده‌سازی شود تا به وسیله آن بتوان تکنیک‌های مختلف برقراری ارتباط با سیستم عامل‌های مختلف را به ابزار اضافه کرد. برای این منظور قابلیت توسعه‌پذیری به ابزار اضافه خواهد شد. برای پیاده‌سازی این قابلیت نیاز است تا معماری نرم‌افزار به این صورت طراحی شود که هسته اصلی سیستم، کتابخانه پردازش فایل پایگاه‌داده باشد و مکانیزم برقراری ارتباط با

[۱۴] System File

[۱۵] Format

سیستم‌عامل‌ها و گوشی‌های مختلف به عنوان افزونه[۱۶] به سیستم اضافه خواهد شد. برای پیاده‌سازی این معماری از زبان C# استفاده خواهد شد. این زبان دارای این قابلیت است که می‌توان فایل‌های dll (برنامه‌های به زبان C# بعد از کامپایل در قالب یک فایل با پسوند dll. ذخیره می‌شوند.)را در زمان اجرا[۱۷] به ابزار اضافه کرد و از تابع‌های آن استفاده کرد.

در این پروژه تاکید بر روی سیستم عامل اندروید خواهد بود. افزونه اتصال به گوشی‌های مبتنی بر این سیستم عامل ارائه خواهد شد که شامل امکان تعریف نام و مسیر نرم‌افزارهایی است که از آنها فایل پایگاه‌داده استخراج خواهد شد و همچنین شامل واحدی است که امکان برقراری ارتباط با گوشی را فراهم می‌آورد.

در ادامه نحوه طراحی و پیاده‌سازی کتابخانه پردازش فایل پایگاه‌داده Sqlite و افزونه اندروید و قابلیت توسعه‌پذیری به تفصیل بیان خواهد شد.

۵.۱ ساختار پایان‌نامه

در ادامه‌ی پایان‌نامه در فصل دوم به طور مختصر گوشی‌های تلفن همراه اندرویدی مورد بررسی قرار می‌گیرند. در این فصل به صورت اجمالی سیستم عامل اندروید معرفی می‌شود. سپس نحوه استفاده از ADB shell بیان خواهد شد و ابزاری برای دستیابی به سطح دسترسی کاربر ممتاز ارائه می‌شود.

در فصل سوم پایگاه‌داده Sqlite و ساختار آن مورد بررسی قرار می گیرد. در این فصل به تفصیل قالب فایل این پایگاه‌داده و نواحی حائز اهمیت برای بازیابی اطلاعات مورد بررسی قرار خواهد گرفت.

در فصل چهارم روش‌ها و الگوریتم‌های بازیابی اطلاعات پاک‌شده و تغییریافته بیان می‌شود. در این فصل با تکیه بر توضیحات ارائه شده در فصل سوم تکنیک‌های بازیابی اطلاعات از فایل اصلی پایگاه‌داده و فایل جانبی آن (فایل ژورنال) مورد بررسی قرار خواهد گرفت.

---

[۱۶] Plugin

[۱۷] Run time

در فصل پنجم نحوه پیاده‌سازی نرم‌افزار و تحلیل و طراحی آن، نمودارهای تحلیل و اجرای قابلیت توسعه‌پذیری و همچنین نمایی از رابط گرافیکی نرم‌افزار بررسی می‌شوند. سرانجام در فصل ششم به جمع‌بندی و کارهای آینده پرداخته خواهد شد.

۶.۱ جمع‌بندی

در این فصل به طور اجمالی به تعریف مسئله و اهمیت موضوع پرداخته شد. همان طور که ذکر شد، هدف از این پروژه پیاده‌سازی ابزاری با قابلیت توسعه‌پذیری برای استخراج اطلاعات از گوشی‌های تلفن همراه هوشمند است. علت اهمیت اطلاعات ذخیره شده در گوشی‌های هوشمند و دلیل اهمیت داده‌های پاک‌شده مورد بررسی قرار گرفت. همچنین بیان شد برای این که ابزار پیاده‌سازی شده، گوشی‌ها و سیستم عامل‌های گوناگون را پشتیبانی کند، نیاز است تا قابلیت توسعه‌پذیری به سیستم اضافه شود. علاوه بر آن ذکر شد که تاکید پروژه کنونی بر روی سیستم عامل اندروید و بازیابی اطلاعات از پایگاه‌داده Sqlite خواهد بود.

۲

فصل دوم

بررسی اجمالی گوشی‌های تلفن هوشمند مبتنی بر سیستم عامل اندروید

در این فصل هدف معرفی سیستم عامل اندروید و نحوه پیاده‌سازی نرم‌افزارهای اندرویدی است. همچنین جایگاه پایگاه‌داده Sqlite در نرم‌افزارها مورد بررسی قرار می‌گیرد. در ادامه نحوه برقراری ارتباط با سیستم عامل و دریافت فایل پایگاه‌داده بررسی می‌شود.

۱.۲ سیستم عامل اندروید

شکل ۱ لوگوی اندروید

همان طور که در [۲] آمده است اندروید در یونانی به معنای مرد، انسان، شبه آدم یا ربات، نام سیستم عاملی است که گوگل برای تلفن‌های هوشمند و تبلت‌ها و هم اکنون برای تلویزیون‌ها عرضه می‌نماید و با همکاری ده‌ها شرکت بر روی دستگاه‌های مبتنی بر اندروید قرار می‌دهد. اندروید بر پایه هسته لینوکس ساخته شده است.

در اوت ۲۰۰۵، گوگل شرکت اندروید واقع در پالو آلتو، کالیفرنیا را خرید. شرکت کوچک اندروید که توسط اندی رابین، ریچ ماینرز، نیک سیرز و کریس وایت پایه‌گذاری شده بود، در زمینه تولید نرم‌افزار و برنامه‌های کاربردی برای تلفن‌های همراه فعالیت می‌کرد. اندی رابین مدیر عامل اجرایی این شرکت پس از پیوستن اندروید به گوگل به سمت قائم‌مقام مدیریت مهندسی این شرکت و مسئول پروژه اندروید در گوگل منصوب شد.

نخستین گوشی مبتنی بر اندروید توسط شرکت اچ‌تی‌سی با همکاری تی-موبایل تولید شد. این گوشی به فاصله کمتر از یک سال از تشکیل اتحادیه گوشی باز یعنی در ۲۲ اکتبر ۲۰۰۸ تولید شد. در ۳ سپتامبر ۲۰۱۳ توسعه‌دهندگان اندروید به‌طور رسمی اعلام کردند که با شرکت نستله، که از شرکت‌های مطرح صنعت شکلات‌سازی جهان می‌باشد، همکاری خواهند کرد. در همین راستا نگارش ۴,۴ سیستم‌عامل

اندروید، کیت‌کت نام گرفت .کیت کت از مارک‌های معروف شکلات است که توسط شرکت نستله تولید می‌شود. آخرین نسخه اندروید، یعنی نسخه ۵,۰، نیز ۵ جولای ۲۰۱۴ با نام آب‌نبات چوبی عرضه شده است.

۲.۲ معماری سیستم عامل اندروید

شکل ۲ صفحه خانگی اندروید نسخه ۵,۰

در [۲] آمده‌است که هسته اصلی سیستم عامل اندروید، سیستم عامل لینوکس[۱۸] نسخه ۲,۶ است. وظیفه اصلی هسته آن راه‌اندازی[۱۹] و مدیریت سخت‌افزار و نرم‌افزارهای موجود بر روی سیستم عامل را بر عهده دارد. همان طور که در شکل ۳ دیده می‌شود، تمام کاربردهای سطح پایین مثل راه‌اندازی و مدیریت دوربین، صفحه نمایش و غیره بر عهده هسته سیستم عامل خواهد بود. در لایه بالاتر مجموعه‌ای از کتابخانه‌ها قرار گرفته است که برای توسعه‌دهندگان[۲۰] نرم‌افزارهای تحت اندروید مورد استفاده قرار می‌گیرد. در شکل تعدادی از این کتابخانه‌ها نمایش داده شده است. مهمترین این کتابخانه‌ها، کتابخانه SQLite است که از نظر بازیابی و استخراج اطلاعات نرم‌افزارها حائز اهمیت است.

نرم‌افزارهای تحت اندروید با استفاده از کتابخانه‌های هسته[۲۱] و JVM[۲۲] اجرا می‌شوند. در نهایت SDK[۲۳] به واسطه چارچوب[۲۴] و API هایی که تدارک دیده است این امکان را برای توسعه‌دهندگان فراهم می‌آورد تا با امکانات سطح پایین سیستم عامل ارتباط برقرار کند و از آنها استفاده کند.

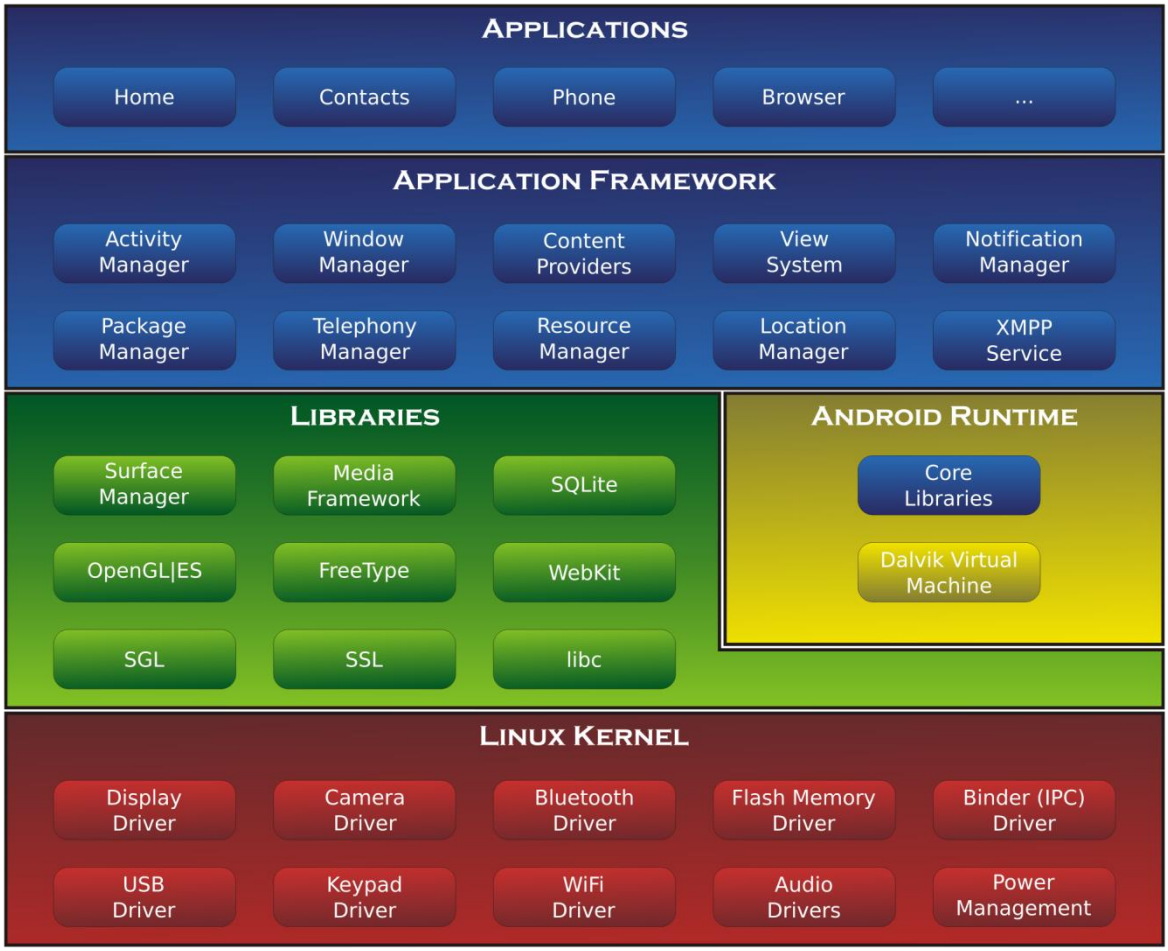
---

[۱۸] Linux

[۱۹] Boot

[۲۰] Developers

[۲۱] Core Libraries



شکل ۳ نمایی از معماری سیستم عامل اندروید[۲]

برای اینکه سیستم به طور بهینه از منابع خود یعنی پردازنده، حافظـه اصـلی [۲۵] و بـاتری اسـتفاده کند، برنامه‌های تحت سیستم عامل اندروید تحت Dalvik VM اجرا می‌شوند. برای اینکه خـطِ مشـی [۲۶] های مورد نظر اندروید اعمال شوند و برنامه‌ها در محیطی امن اجرا شوند، سیستم عامل اندروید برای هر برنامه از Dalvik VM جداگانه استفاده می‌کند. Dalvik VM برای اینکه بتواند استفاده بهینه از منابع را

---

[۲۲] Java Virtual Machine

[۲۳] Software Development Kit

[۲۴] Frame work

[۲۵] RAM

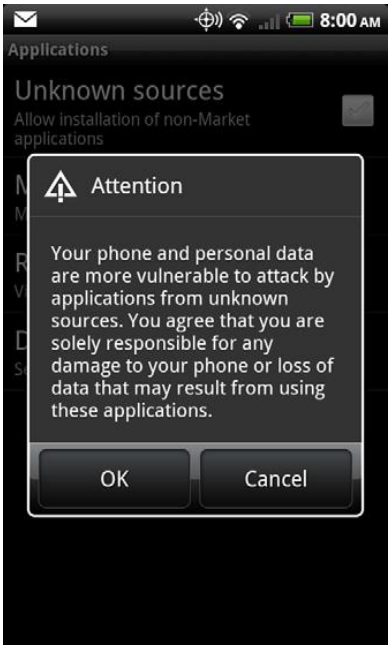
[۲۶] Policy

تضمین کند به هسته لینوکس بسیار وابسته است. برنامه‌های تحت سیستم عامل اندروید بـه زبـان جـاوا نوشته و کامپایل می‌شوند. سپس بایت‌کدهای تولید شده به فرمت خاصی به نام dex تبدیل می‌شوند که برای اجرای بهینه تحت Dalvik VM فراهم آمده‌اند.

۳.۲ مدل امنیتی اندروید

نرم‌افزارهای تحت اندروید دارای پسوند apk هستند. هر فایل با این پسوند توسـط سیسـتم عامـل چک می‌شود تا دارای امضای معتبر باشد و در صورت اعتبار به کاربر اجازه نصب نرم‌افزار را می‌دهد. بعـد از اطمینان از اعتبار امضا، دسترسی‌های نرم‌افزار بررسی شده در اختیار کاربر قرار می‌گیرد تا کاربر اعتبار این دسترسی‌ها را تایید کند. برای مثال یک نرم‌افزار ممکن است به نـرم‌افـزار مخاطبـان یـا پیـام‌رسـان دسترسی داشته باشد، در این حالت سیستم عامل این دسترسی‌ها را به کاربر هشدار می‌دهـد وکـاربر در صورت تایید اجازه نصب نرم‌افزار را می‌دهد.

همان طور که در شکل ۴ آمده است در مورد امضای معتبر، اندروید دارای این امکان است کـه بـا فعال کردن آن می‌توان نرم‌افزارها را از منبعی به غیر از Play Store (بازار نرم‌افزارهـای اندرویـدی تحـت



شکل ۴ تنظیمات اندروید برای نصب نرم‌افزار از منبع نا شناس

نظارت گوگل) دانلود و نصب کرد.

علاوه بر موارد بالا برای هر نرم‌افزار در هسته سیستم عامل یک پردازه[۲۷] اجرا می‌شود و چون که هسته سیستم بر مبنای لینوکس است به هر کدام از پردازه‌ها شناسه کاربر[۲۸] و شناسه گروه[۲۹] اختصاص داده می‌شود. پس هر نرم‌افزار تحت Dalvik VM و پردازه مختص به خود اجرا می‌شود. برای اعمال خطِ مشی امنیتی در سطح سیستم عامل برای تضمین جداسازی حافظه به هر نرم‌افزار برای ذخیره‌سازی داده‌های خود، دایرکتوری جداگانه اختصاص داده می‌شود که از دید نرم‌افزارهای دیگر پنهان است.

۴.۲ اتصال گوشی‌های مبتنی بر اندروید به رایانه

SDK تنها ابزاری برای برنامه‌نویسی و تولید نرم‌افزارهای تحت اندروید نیست. علاوه بر آن امکانات دیگری را نیز فراهم می‌کند که به برنامه‌نویس در این فرایند کمک می‌کند. یکی از این ابزارها که در استخراج فایل‌های پایگاه‌داده به ما کمک می‌کند ابزار ADB[۳۰] است. این ابزار این امکان را می‌دهد تا بتوان از طریق رایانه به پایانه سیستم عامل اندروید متصل شد و دستورات لازم را اجرا کرد که در ادامه به بررسی دستوراتی که برای پیاده‌سازی این ابزار به آنها نیاز است پرداخته خواهد شد.

پیش از بررسی دستورات، باید تنظیمات لازم برای اتصال گوشی به رایانه را در گوشی انجام داد. برای این کار باید در مسیر Settings\Developer Options، USB Debugging را فعال کرد. در نسخه‌های جدید اندروید قسمت Developer Option پنهان است که با چندین بار انتخاب پشت سر هم Build number در مسیر Settings\About Phone می‌توان آن را فعال کرد. همان طور که در شکل ۵ آمده است بعد از فعال شدن، هشداری مبنی بر فعال شدن Developer Option پدیدار می‌شود. در شکل ۶ فعال کردن USB Debugging آمده است.

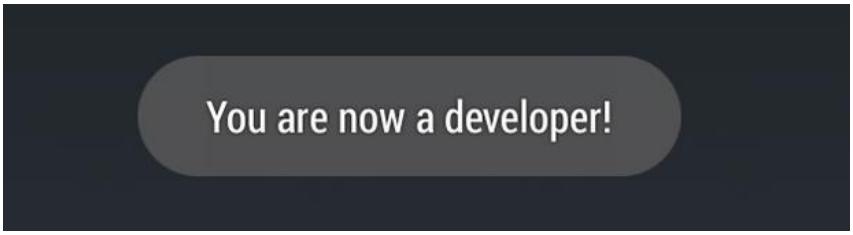
---

[۲۷] Process

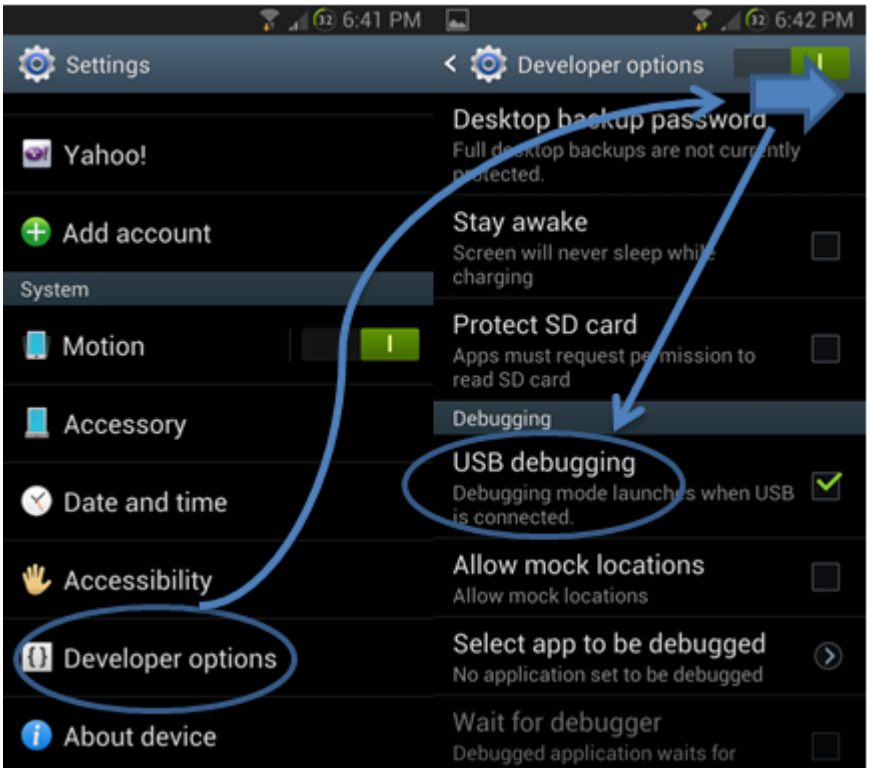
[۲۸] User ID

[۲۹] Group ID

[۳۰] Android Debug Bridge



شکل ۶ هشدار سیستم عامل مبنی بر فعال شدن Developer Option



شکل ۵ فعال کردن USB Debugging

همان طور که در [۳] آمده است، محل ذخیره‌سازی ابزار ADB در <SDK_Path>\platform-tools\adb.exe است. از طریق دستوراتی که این ترمینال فراهم می‌آورد، می‌توان عملیات‌های مورد نیاز از قبیل تغییر تنظیمات دسترسی به فایل‌ها، کپی کردن فایل‌ها از گوشی به رایانه و نصب یک نرم‌افزار روی گوشی را انجام داد. که در جدول شماره ۱ چند مورد از دستورات آورده شده است.

دسترسی به shell اندروید	adb shell
کپی کردن فایل از گوشی به رایانه	adb push <path in pc> <path in phone>
نصب example.apk از رایانه	adb install example.apk
لیست گوشی‌های متصل	adb devices

جدول ۱ دستورات adb shell

۵.۲ نحوه ذخیره‌سازی اطلاعات نرم‌افزارها در اندروید

در [۲] آمده است که اطلاعات ذخیره‌سازی شده هر نرم‌افزار در سیستم عامل اندروید را می‌توان از جنبه ذخیره‌سازی روی حافظه داخلی یا خارجی بررسی کرد. نرم‌افزارها در حافظه خارجی بدون محدودیت می‌توانند اطلاعات خود را ذخیره‌سازی کنند ولی در حافظه داخلی باید در مسیر \data\data\<pkg_name> ذخیره شوند. دایرکتوری‌هایی که در ذیل این مسیر می‌آیند و کاربرد آنها در جدول ۲ آمده است.

دایرکتوری ذخیره‌سازی فایل‌های shared prefrencess	shared_prefs
کتابخانه‌هایی که به صورت موردی در نرم‌افزار نیاز است	Lib
فایل‌هایی که برنامه‌نویس در حافظه داخلی ذخیره می‌کند	Files
فایل‌هایی که توسط نرم‌افزار در حافظه داخلی کش می‌شوند.	Cache
فایل‌های sqlite و فایل Journal	Databases

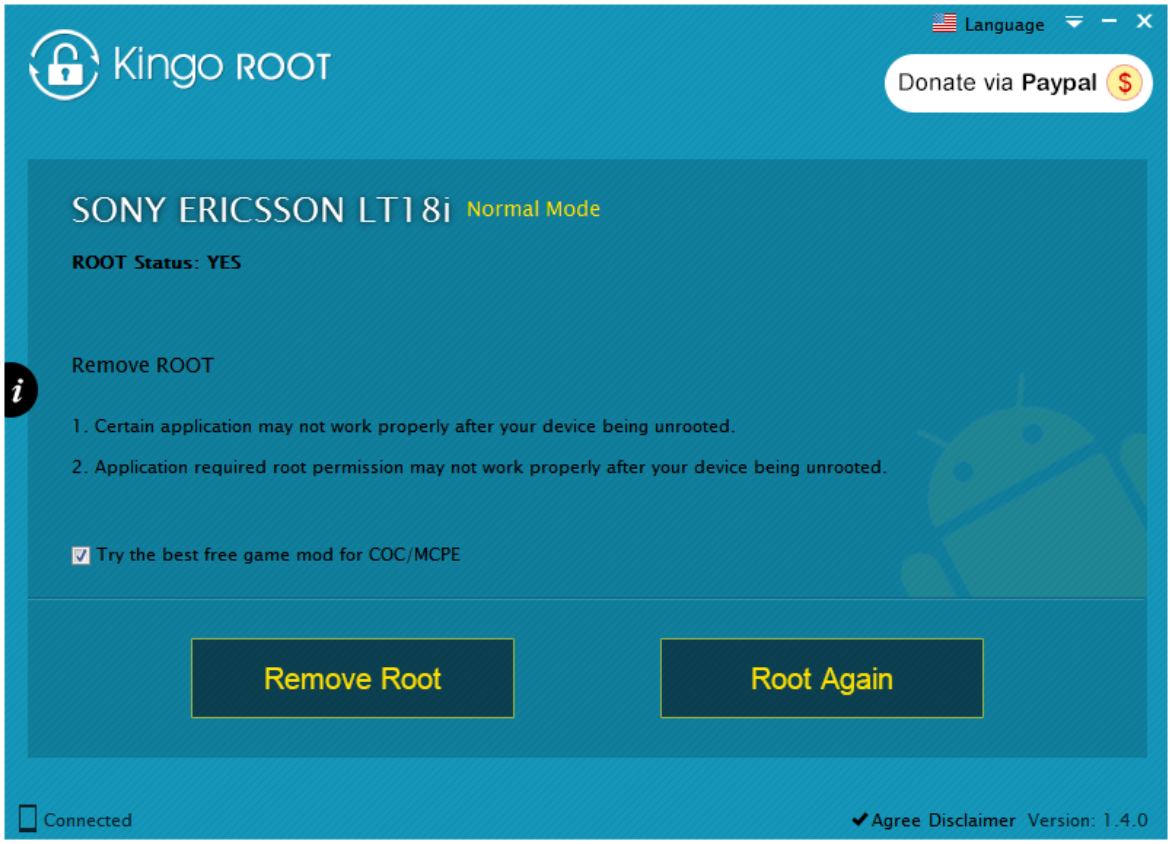
جدول ۲ دایرکتوری‌های موجود در نرم‌افزارهای تحت اندروید

در این پروژه هدف دستیابی به فایل‌های پایگاه‌داده Sqlite است. برای دسترسی به این فایل‌ها و تغییر تنظیمات مربوط به سطح دسترسی به این فایل‌ها نیاز است که کاربر ممتاز[۳۱] این تغییرات را انجام

[۳۱] Privileged user

دهد. دستیابی به سطح کاربری ممتاز را اصطلاحا روت[۳۲] کردن سیستم گویند. پس اولین قدم برای دستیابی به فایل‌های پایگاه‌داده، روت کردن سیستم است.

در این پروژه برای روت کردن سیستم از ابزار Kingo Root[۳۳] استفاده شده‌است. این ابزار قادر است سیستم‌های اندرویدی با پلتفرم‌های مختلف و سیستم عامل‌های اندروید با نسخه‌های مختلف را روت کند. این ابزار بروی سیستم عامل ویندوز و اندروید ارائه شده‌است. نمایی از این ابزار در شکل‌های ۷و۸ قابل مشاهده هستند. مهمترین مزیت این نرم‌افزار نسبت به نمونه‌های مشابه این است که بعد از



شکل ۷ نمایی از ابزار Kingo Root تحت ویندوز

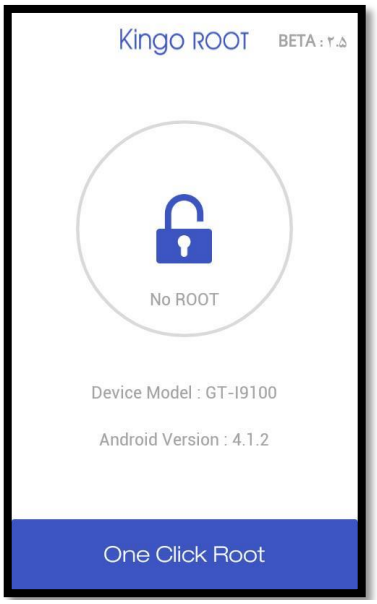
اتصال گوشی به رایانه – با روشی که در بالا ذکر شد– فقط با فشردن کلید روت، ابزار عملیات لازم برای روت کردن گوشی را انجام می‌دهد. در حالی که در ابزارهای مشابه نیاز است تا کلیدهای روی گوشی فشرده شوند یا گوشی به طور دستی دوباره راه‌اندازی شود.

---

[۳۲] Root

[۳۳] http://www.kingoapp.com/

بعد از روت کردن سیستم، از طریق دستور chmod 777 <path> دسترسی به فایل‌های پایگاه‌داده تنظیم می‌شود. سپس از طریق دستور push فایل‌ها از گوشی به رایانه کپی می‌شوند. برای اجرای این دستورات کتابخانه Android lib استفاده‌شد. این کتابخانه دستوراتی که در نرم‌افزار ADB فراهم آمده است (مثل دستورات push یا install به جدول ۱ مراجعه شود.) را به زبان C# پیاده‌سازی کرده‌است. از این کتابخانه در افزونه[۳۴] اندروید استفاده می‌شود که در فصل پنجم به تفصیل چگونگی استفاده و پیاده‌سازی این افزونه بررسی خواهد شد.



شکل ۸ نمایی از ابزار Kingo Root تحت اندروید

[۳۴] Plugin

۶.۲ جمع‌بندی

در این فصل به طور اجمالی سیستم عامل اندروید مورد بررسی قرار گرفت. ابتدا تاریخچه‌ای کوتاه و همچنین نسخه‌های مختلف آن بررسی شد. در ادامه معماری این سیستم عامل، نحوه پیاده‌سازی و اجرای نرم‌افزارهای مختلف تحت اندروید بررسی شد. سپس مدل امنیتی اندروید و نحوه پنهان سازی اطلاعات هر نرم‌افزار از دیگری مورد بررسی قرار گرفت. در پایان نحوه اتصال گوشی به رایانه و استفاده از ابزارهای ADB و Kingo Root بیان شدند.

۳

فصل سوم

بررسی ساختار پایگاه‌داده Sqlite

در این فصل به صورت اجمالی ساختار پایگاه‌داده Sqlite مورد بررسی قرار می‌گیرد. پایگاه‌داده Sqlite شامل یک فایل اصلی و یک فایل ژورنال[۳۵] است که در ادامه ساختار این دو فایل بررسی می‌شوند. مطالب عنوان شده در این فصل بر گرفته از [۴] و [۵] است.

۱.۳ ساختار فایل اصلی پایگاه‌داده Sqlite

۱.۱.۳ صفحه‌ها[۳۶]

فایل اصلی پایگاه‌داده شامل تعدادی از صفحه‌هاست. اندازه هر صفحه توانی از دو، میان ۵۱۲ تا ۶۵۵۳۶ بایت می‌تواند باشد. اندازه صفحه‌ها در هر پایگاه داده در ۲ بایت در مکان[۳۷] ۱۶ از سرآیند[۳۸] فایل اصلی پایگاه‌داده قرار می‌گیرد. هر یک از این صفحه‌ها در پایگاه‌داده دارای نقشی هستند که با توجه به اینکه در این پروژه فقط از یکی از این نقش‌ها استفاده شده است و آن صفحات موجود در B-Tree جدولی پایگاه‌داده است، این گونه صفحات در ادامه بررسی می‌شوند.

---

[۳۵] Journal file

[۳۶] Pages

[۳۷] Offset

[۳۸] Header

۲.۱.۳ سرآیند فایل اصلی پایگاه‌داده

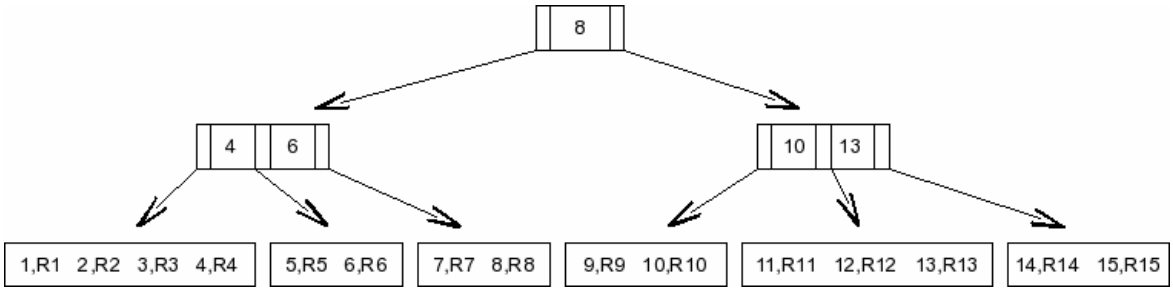
فایل اصلی پایگاه‌داده دارای ۱۰۰ بایت سرآیند است. با توجه به کاربرد این سرآیندها در پروژه حاضر، تعدادی از این سرآیندها در جدول ۳ بررسی شده‌اند.

مکان	اندازه	توضیحات
۰	۱۶	برای تشخیص فایل‌های پایگاه‌داده Sqlite در ابتدای فایل، رشته "SQLite format 3\000" وجود دارد.
۱۶	۲	اندازه صفحه‌ها در پایگاه‌داده

جدول ۳ سرآیندهای مورد استفاده فایل اصلی پایگاه‌داده [۴]

۳.۱.۳ صفحات B-Tree جدولی

جداول موجود در پایگاه‌داده در ساختار B-Tree ذخیره شده‌اند. شماره صفحه ریشه در جدول sqlite_master ذخیره شده است. ساختار صفحات در B-Tree در ادامه ذکر شده‌است. ساختار صفحات میانی[۳۹] و برگ[۴۰] مشابه هم هستند با اندکی تفاوت، که در ادامه بررسی می‌شوند.



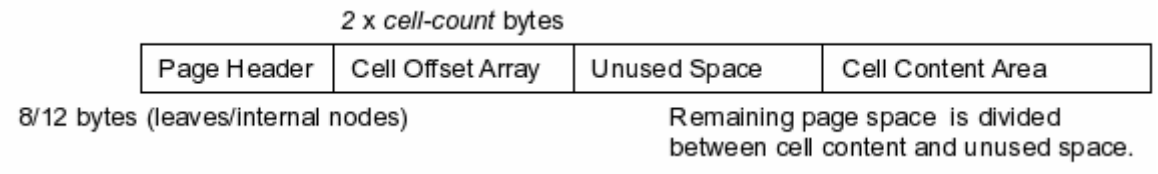
شکل ۹ نمایش جداول پایگاه‌داده در ساختار B-Tree [۵]

[۳۹] Internal

[۴۰] Leaf

۱.۳.۱.۳ ساختار صفحات B-Tree جدولی

همان‌طور که در شکل ۱۰ مشاهده می‌شود صفحات در B-Tree جدولی از چهار بخش تشکیل شده‌اند: سرآیند صفحه، آرایه مکان‌های سلول‌ها، فضای خالی و فضای قرار گیری سلول‌ها. سرآیندهای صفحات در جدول ۴ آمده است.



شکل ۱۰ ساختار صفحات B-Tree جدولی [۵]

مکان	اندازه	توضیحات
۰	۱	پرچم صفحه B-Tree
۱	۲	مکان اولین بلوک در لیست بلوک‌های فضای آزاد. در صورت صفر بودن لیست خالی است.
۳	۲	تعداد سلول‌های موجود در صفحه
۵	۲	مکان بایت اول فضای سلول‌ها
۷	۱	تعداد بایت‌های آزاد پراکنده[۴۱] در صفحه
۸	۴	شماره صفحه راست‌ترین اشاره‌گر در درخت

جدول ۴ مشخصات سرآیندهای موجود در صفحه‌های B-Tree جدولی [۴]

با توجه به توضیحات موجود در جدول ۴ ذکر دو نکته لازم است:

- اول اینکه فضاهای آزاد میان فضای در حال استفاده، اگر حجم کمتر مساوی ۳ بایت داشته باشند به عنوان بایت‌های آزاد پراکنده تلقی می‌شوند.
- دوم اینکه اگر این فضاهای آزاد بیشتر از ۳ بایت باشند، به عنوان بلوک‌های آزاد[۴۲] در نظر گرفته می‌شوند و در یک لیست پیوندی قرار می گیرند. مکان اولین بلوک در مکان ۱ از سرآیند صفحه

[۴۱] Fragmented free spaces

قرار می گیرد. در هر گره دو بایت اول، مکان گره بعد نسبت به مکان شروع صفحه و دو بایت دوم اندازه بلوک را مشخص می‌کنند. در صورت صفر بودن دو بایت اول، گره کنونی گره آخر خواهد بود.

۲.۳.۱.۳ تفاوت صفحه‌های جدولی داخلی و برگ

- در صفحه‌های داخلی بایت پرچم ۵ و در صفحه‌های برگ بایت پرچم ۱۳ است.
- ساختار و محتوای سلول‌ها متفاوت است.

ساختار سلول‌ها در صفحه‌های داخلی در شکل ۱۱ آمده‌است. ۴بایت اول شماره صفحه فرزند[۴۳] است قسمت دوم سلول از نوع Var Int[۴۴] است که به عنوان کلید سلول تلقی می‌شود. برای اینکه بتوان عددی با مقدارهای زیاد(long type) را به صورت بهینه ذخیره کرد از ساختار Var Int استفاده می‌شود. در این ساختار که حداکثر ۹ بایت می‌تواند باشد، هر بایت به غیر از بایت نهم دارای ۷ بیت داده و یک بیت پرچم است. که سمت چپ‌ترین بیت، بیت پرچم خواهد بود. در صورتی که پرچم یک باشد بایت بعدی هم جز عدد خواهد بود در غیر این صورت بایت کنونی بایت آخر است.

Child page number	Integer Value
4 bytes	1-9 bytes

شکل ۱۱ ساختار سلول‌های صفحه‌های داخلی B-Tree جدولی [۵]

در مورد صفحات برگ ساختار سلول بستگی به اندازه اطلاعات دارد که در صورت کوتاه بودن شکل ۱۲ و در صورت طولانی بودن شکل ۱۳ خواهد بود. در صورت طولانی بودن، اطلاعات در قالب لیست پیوندی ذخیره می‌شوند. که ساختار هر گره میانی در لیست، در شکل ۱۴ آمده است. طولانی بودن رکوردها از رابطه زیر محاسبه می‌شوند:

min-local := (*usable-size* - 12) * *min-embedded-fraction* / 255 – 23
max-local := *usable-size* – 35
local-size := *min-local* + (*record-size* - *min-local*) % (*usable-size* - 4)

---

[۴۲] Free Blocks

[۴۳] Child

[۴۴] Variable length integer توضیحات مربوط به این نوع ساختار در مرجع [4] آمده است.

if(*local-size* > *max-local*)
local-size := *min-local*

مقدار min-embedded-fraction در مکان ۲۲ از سرآیند فایل اصلی قابل محاسبه است. مقـدار local-size تعدادبایت‌هایی است که در صفحه B-Tree ذخیره می‌شود. در صورتی که رکورد از این مقدار بزرگتر باشد، ادامه رکورد در یک لیست پیوندی ذخیره می‌شود. همان طور که در شکل ۱۳ آمده اسـت، ۴ بایت آخر سلول، شماره صفحه حاوی ادامه داده‌های رکورد می‌آید. ساختار صفحه‌های بعدی در شـکل ۱۴ آمده است که ۴ بایت اول شماره صفحه بعدی است. در صورت صفر بودن، صفحه کنونی صفحه آخر خواهد بود.

Record Size	Key Value	Database Record
1-9 bytes	1-9 bytes	*Record-size* bytes

شکل ۱۲ ساختار سلول‌ها با داده‌های کوتاه در صفحات برگ B-Tree جدولی [۵]

Record Size	Key Value	Database Record Prefix	Overflow page number
1-9 bytes	1-9 bytes	*local-size* bytes, where *local-size* is as defined above	4 bytes

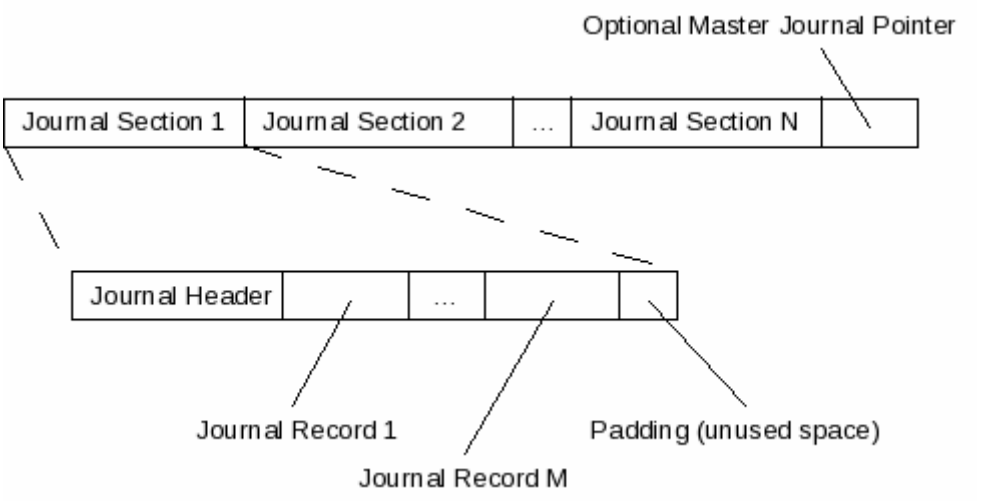
شکل ۱۳ ساختار سلول‌ها با داده‌های طولانی در صفحات برگ B-Tree جدولی [۵]

Next Overflow Page number	Record Data
4 bytes	Remaining space

شکل ۱۴ ساختار گره‌های میانی در لیست‌پیوندی مربوط به داده‌های طولانی در صفحات برگ B-Tree جدولی [۵]

۲.۳ ساختار فایل ژورنال

در پایگاه‌داده Sqlite علاوه بر فایل اصلی پایگاه‌داده، فایل دیگری موسوم به فایل ژورنال[۴۵] استفاده می‌شود. نام فایل ژورنال با اضافه شدن رشته "jornal-" به اسم فایل اصلی ایجاد می‌شود. هنگام اجـرای هر تراکنش[۴۶] صفحه‌ای که تغییرات در آن اعمال می‌شود، به همراه شماره آن در فایـل اصـلی، در فایـل ژورنال کاملا کپی می‌شود تا در صورت ایجاد هر مشکلی اطلاعات قابل بازگشت باشند. ساختار کلی فایل ژورنال در شکل ۱۵ آمده است. در ادامه ساختار سرآیندها و رکورد[۴۷]هـا در شـکل هـای ۱۶ و ۱۷ آمـده است.

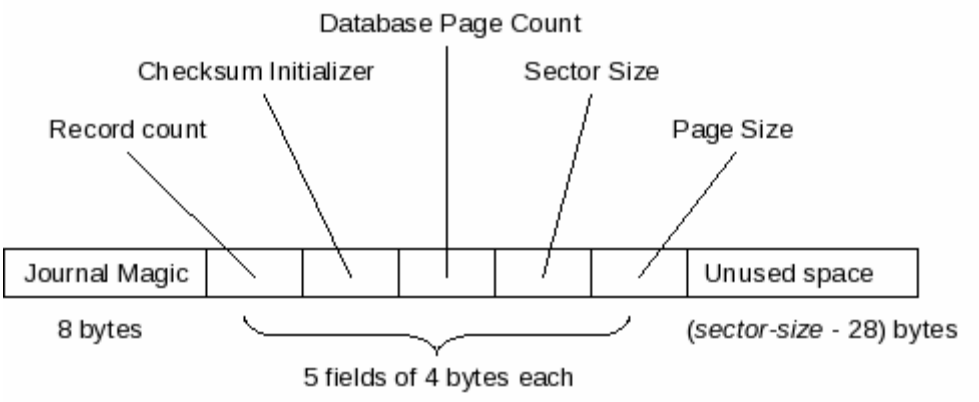


شکل ۱۵ ساختار کلی فایل ژورنال [۵]

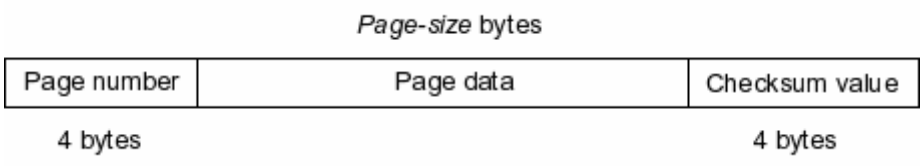
[۴۵] Journal

[۴۶] Transaction

[۴۷] Record



شکل ۱۷ ساختار سرآیندهای فایل ژورنال [۵]



شکل ۱۶ ساختار رکوردهای فایل ژورنال [۵]

در صورتی که صفحه‌ای در فایل ژورنال معتبر باشد، یعنی تغییرات در فایل اصلی اعمال نشده باشد، سرآیندهای فایل ژورنال معتبر خواهد بود. در صورت عدم وجود چنین صفحه‌ای پایگاه‌داده تمام بایت‌های موجود در سرآیند را صفر خواهد کرد. از جمله قسمت‌های مهم در سرآیند فایل ژورنال قسمت Journal magic است که از آن برای شناسایی فایل‌های ژورنال استفاده می‌شود. دیگری اندازه صفحه‌هاست که در روش سوم بازیابی مورد استفاده قرار می‌گیرد. علاوه بر آن اندازه سکتورهاست که مقدار پیش‌فرض آن ۵۱۲ بایت است. ساختار فایل اینگونه است که فایل از تعدادی سکتور تشکیل شده است. اگر اندازه سکتور به اندازه رکوردها بخش‌پذیر نباشد، فضای باقی مانده به عنوان فضای بلااستفاده تلقی می‌شود. در خواندن سکتورها باید به این فضاها توجه شود.

اگر بخواهیم ساختار فایل ژورنال را جمع بندی کنیم، می توان گفت که بعد از ۲۸ بایت سرآیند، رکوردها قرار می‌گیرند که خود شامل شماره صفحه، صفحه کپی شده از فایل اصلی و در ادامه ۴ بایت checksum است.

۳.۳ جمع‌بندی

در این فصل ساختار فایل‌های پایگاه‌داده Sqlite مورد بررسی قرار گرفت. همان طور که بیان شد این پایگاه‌داده از دو فایل اصلی و فایل ژورنال تشکیل شده‌اند. بررسی ساختار فایل اصلی نشان می‌دهد که این پایگاه‌داده بعد از پاک‌کردن رکوردها، اطلاعات را پاک نمی‌کند بلکه نشانگر به ابتدای آن رکورد از پایگاه‌داده پاک می‌شود. پس از این نکته می‌توان برای بازیابی اطلاعات از فایل اصلی استفاده کرد. همچنین در فایل ژورنال بعد از اتمام تراکنش، صفحه یا صفحاتی که برای پشتیبانی در این فایل ذخیره شده‌اند، پاک نمی‌شوند بلکه سرآیند این فایل به نحوی که نشان دهد این صفحات نامعتبرند، تغییر می‌کند. پس از اطلاعات باقی‌مانده در این فایل نیز برای بازیابی می‌توان استفاده کرد. که روش‌های بازیابی در فصل بعد به تفصیل بیان خواهد شد.

۴

فصل چهارم

روش‌های بازیابی اطلاعات پاک‌شده و تغییریافته از پایگاه‌داده Sqlite

در این فصل، ۳ روش برای بازیابی اطلاعات پاک‌شده و تغییریافته ارائه می‌شود که ۲ روش بر اساس فایل اصلی و یک روش بر اساس فایل ژورنال است.

۱.۴ بازیابی بر اساس فایل اصلی پایگاه‌داده

با توجه به ساختار پایگاه‌داده Sqlite که در فصل گذشته بررسی شد، می‌توان به این نتیجه رسید که می‌توان ۲ روش برای بازیابی اطلاعات پاک‌شده یا تغییر پیدا کرده پیشنهاد کرد.

در پایگاه‌داده Sqlite در فایل اصلی، هنگامی که رکوردی پاک می‌شود اشاره‌گر به سلول حاوی این رکورد از آرایه اشاره‌گرها در صفحه مربوط به آن، پاک می‌شود. اطلاعات موجود در این رکورد، از این به بعد از نظر پایگاه‌داده معتبر نخواهد بود و پایگاه‌داده از آن به عنوان فضای آزاد استفاده می‌کند.

۱.۱.۴ پیمایش B-Tree جدولی

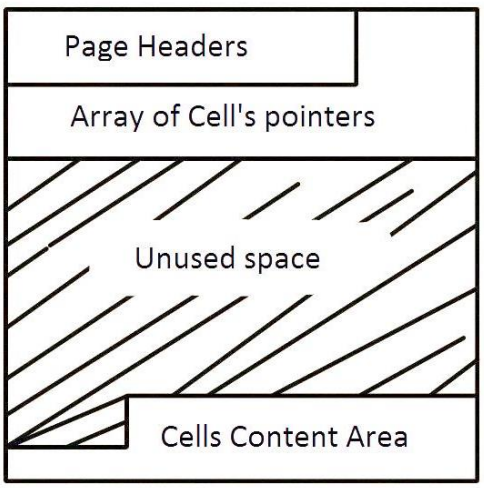
برای بازیابی اطلاعات جداول پایگاه داده لازم است تا B-Tree شامل این جدول پیمایش شود. ابتدا باید شماره صفحه ریشه این درخت استخراج شود. برای این کار از طریق جدول sqlite_master شماره صفحه ریشه استخراج می‌شود. سپس از طریق رابطه page_offset=(page_number - 1)*page_size مکان صفحه ریشه بدست می‌آید.

همان طور که در فصل پیش ذکر شد صفحات موجود در یک درخت دو نوع دارند: صفحات داخلی و صفحات برگ، که باتوجه به پرچم قابل تفکیک هستند. برای رسیدن به

صفحات برگ، با شروع از ریشه و با استفاده از یک تابع بازگشتی صفحات داخلی پیمایش می‌شوند. سلول های صفحات داخلی حاوی شماره صفحات فرزند خود هستند و مانند اشاره‌گر به آنها عمل می‌کنند. با رسیدن به صفحات برگ می‌توان دو روش برای بازیابی رکوردهای پاک‌شده پیشنهاد کرد که در ادامه مورد بررسی قرار می گیرند.

۲.۱.۴ بازیابی از طریق فضای بلااستفاده

با توجه به شکل ۱۸ فضای بلااستفاده که هاشور خورده است می‌تواند مستعد حضور داده‌های پیشین باشد. این فضا میان آرایه اشاره‌گرها به سلول‌ها و فضای شامل سلول‌ها قرار دارد و صفحات با پرچم ۱۳ دارای سلول‌های محتوی داده هستند. همچنین در صورت پاک شدن یک رکورد از پایگاه‌داده، اشاره‌گر سلول آن از آرایه حذف می‌شود و در صورتی که سلول حاوی آن در ابتدای فضای سلول‌ها باشد، اطلاعات آن به انتهای فضای بلااستفاده اضافه می‌شود. پس می‌توان گفت با استخراج فضاهای بلااستفاده می‌توان یک سری از داده‌های پیشین را بازیابی کرد. برای این کار باید B-Tree حاوی هر جدول را پیمایش کرد تا با رسیدن به برگ‌ها بتوان فضای بلااستفاده را استخراج کرد. در این ابزار این فضاها با نام فضاهای تخصیص‌نیافته[۴۸] یادشده‌اند. همچنین داده‌های پاک‌شده قابل تفکیک بر اساس ستون‌های



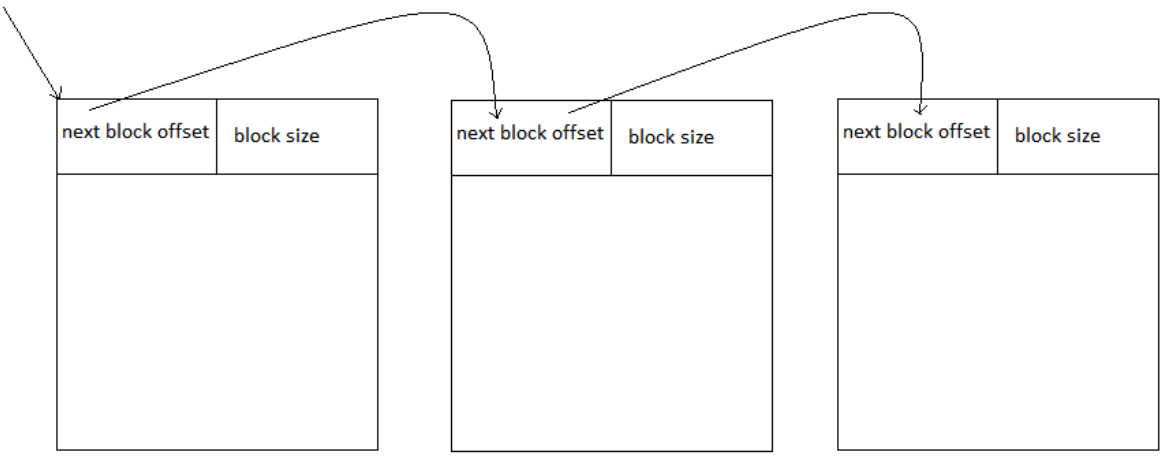
شکل ۱۸ ساختار صفحات B-Tree جدولی [۵]

[۴۸] Unallocated spaces

جدول نیستند چرا که اشاره‌گر به ابتدای سلول پاک شده است. همچنین ممکن است بر روی این سلول‌ها دوباره داده جدید نوشته شده باشد به همین دلیل امکان بازیابی به تفکیک ستون‌ها وجود ندارد. در این پروژه بایت‌هایی که نمایش اسکی دارند از این فضاها استخراج شده و نمایش داده می‌شوند. کاربر با مقایسه این داده‌ها با داده‌های موجود در پایگاه‌داده می‌تواند برخی از داده‌ها را از دیگران تفکیک کند مثلا متن پیام‌ها را از شماره فرستنده جدا کند. این مشکل در بازیابی به روش لیست بلوک‌های آزاد نیز دیده می‌شود که در ادامه به بررسی آن می‌پردازیم.

۳.۱.۴ بازیابی از طریق لیست بلوک‌های آزاد

در صورتی که سلول حاوی رکوردی که از جدول پاک می‌شود، میان سلول‌های فعال دیگر قرار داشته باشد، فضای آن سلول بعد از پاک‌شدن به عنوان بلوک آزاد تلقی می‌شود. در صورتی که این فضا بیشتر از ۳ بایت باشد (که معمولا در مورد سلول‌های شامل رکوردها همین طور است) همان طور که در فصل گذشته بیان شد، به لیست پیوندی بلوک‌های آزاد اضافه می‌شود. بنابراین محل دوم برای بازیابی اطلاعات، این لیست خواهد بود. با پیمایش B-Tree جدولی و دستیابی به صفحه‌های برگ (با پرچم ۱۳) می توان اشاره‌گر به ابتدای این لیست را دریافت و سپس با توجه به ساختار آن که در فصل گذشته ذکر شد آن را پیمایش کرد و داده‌های آن را استخراج نمود. در شکل ۱۹ نمایی از لیست پیوندی بلوک‌های آزاد آمده است.



شکل ۱۹ نمایی از بلوک‌های آزاد در صفحه‌های برگ

۲.۴ بازیابی بر اساس فایل ژورنال پایگاه‌داده

همان طور که در فصل گذشته کاربرد فایل ژورنال بررسی شد، می توان دریافت که هر تغییری در پایگاه‌داده اعم از حذف یا به روزرسانی باعث می‌شود یک کپی از صفحه مورد تغییر، قبل از تغییر در این فایل کاملا ذخیره شود. این صفحه‌ها بعد از پایان تراکنش از نظر پایگاه‌داده معتبر نیستند ولی برای بازیابی اطلاعات می‌توانند مفید باشند.

برای بازیابی این اطلاعات، فایل ژورنال بر اساس ساختار ذکر شده در فصل گذشته خوانده می‌شود و صفحات با پرچم ۱۳ که صفحات برگ در B-Tree هستند همراه با شماره آنها استخراج می‌شوند. در میان این صفحات ممکن است که صفحات با شماره یکسان چند بار تکرار شده باشند که مکانی از فایل ژورنال که این صفحات در آنها وجود دارد در یک لیست قرار خواهد گرفت. همان طور که در شکل ۱۵ آمده است مکان صفحه‌ها همراه با شماره آنها در ساختار موجود در شکل ذخیره شده است.

حال برای استخراج داده‌ها از این روش استفاده می‌شود که به ازای L که طول بلندترین لیست از مکان‌هاست از فایل پایگاه‌داده کپی ایجاد می‌شود. سپس صفحه‌های موجود در هر ستون در صورت وجود، به طور مثال ستون 1، با توجه به رابطه page_offset = (page_number - 1)* page_size در فایل پایگاه‌داده کپی شده، جایگزین می‌شوند. این روال به ازای تمام ستون‌ها در ساختمان‌داده شکل ۲۰ تکرار شده و فایل‌های جدید از پایگاه‌داده تولید می‌شود.

فایل‌های ایجاد شده از طریق جایگزینی صفحه‌ها در فایل اصلی به وسیله ابزار sqldiff.exe که توسط Sqlite برای مقایسه پایگاه‌داده‌ها ارائه شده است[۴]، با فایل اصلی پایگاه‌داده مقایسه می‌شود. خروجی این ابزار به این شرح است که به تفکیک جدول، پرس وجوهایی[۴۹] که مشخص می‌کند چه رکوردی از پایگاه‌داده تولیدی نسبت به پایگاه‌داده اصلی پاک[۵۰]، به‌روز[۵۱] یا اضافه[۵۲] شده است را در خروجی نشان می‌دهد.

---

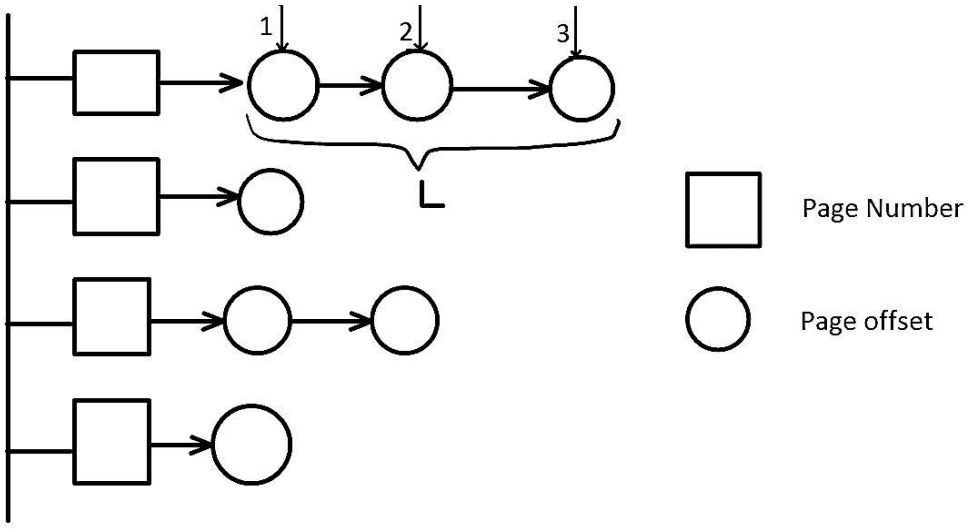
[۴۹] Query

[۵۰] Delete

[۵۱] Update

[۵۲] Insert

پرس‌وجوهای اضافه‌کننده چون در خروجی پایگاه‌داده فعلی قابل مشاهده هستند، مفید نخواهند بود ولی پرس‌وجوهای پاک‌کننده و به‌روزکننده با تغییر به پرس‌وجوی انتخاب[۵۳] از پایگاه داده تولیدی، می تواند رکوردهایی که در پایگاه‌داده فعلی موجود نیست را به ما نشان دهد. شکل ۲۱ خروجی این ابزار را در حالت خلاصه نشان می‌دهد. در نهایت رکوردهای بدست آمده از مقایسه همه فایل‌های تولید شده با فایل اصلی تجمیع شده و در خروجی نشان داده می‌شود.



شکل ۲۰ ساختمان داده استفاده شده برای نگهداری مکان و شماره صفحات موجود در فایل ژورنال

[۵۳] Select

۳.۴ جمع‌بندی

در این فصل الگوریتم‌ها و روش‌های بازیابی اطلاعات از پایگاه‌داده Sqlite مورد بررسی قرار گرفت. همان‌طور که در این فصل بیان شد دو روش برای بازیابی اطلاعات از فایل اصلی پایگاه‌داده و یک روش برای بازیابی از فایل ژورنال بیان شد. همچنین نحوه استفاده از ابزار sqldiff برای مقایسه دو پایگاه‌داده Sqlite مورد بررسی قرار گرفت.

```
addr: 0 changes, 0 inserts, 0 deletes, 0 unchanged
android_metadata: 0 changes, 0 inserts, 0 deletes, 1 unchanged
attachments: 0 changes, 0 inserts, 0 deletes, 0 unchanged
canonical_addresses: 40 changes, 45 inserts, 207 deletes, 0 unchanged
drm: 0 changes, 0 inserts, 0 deletes, 0 unchanged
part: 0 changes, 4 inserts, 0 deletes, 0 unchanged
pdu: incompatible schema
pdu_recipient_threads: missing from first database
pending_msgs: 0 changes, 0 inserts, 0 deletes, 0 unchanged
rate: 0 changes, 0 inserts, 0 deletes, 0 unchanged
raw: 0 changes, 32 inserts, 0 deletes, 0 unchanged
semc_metadata: missing from first database
semc_threads: missing from first database
sms: incompatible schema
sqlite_sequence: 2 changes, 1 inserts, 0 deletes, 0 unchanged
sr_pending: 0 changes, 0 inserts, 0 deletes, 0 unchanged
threads: 35 changes, 52 inserts, 227 deletes, 0 unchanged
words_content: 1576 changes, 81 inserts, 2361 deletes, 0 unchanged
words_segdir: 0 changes, 18 inserts, 28 deletes, 0 unchanged
words_segments: 0 changes, 273 inserts, 114 deletes, 0 unchanged
```

شکل ۲۱ خروجی مقایسه دو پایگاه‌داده که پایگاه‌داده اول پایگاه‌داده حاصل از فایل ژورنال و پایگاه‌داده دوم پایگاه داده اصلی است.

۵

فصل پنجم

پیاده‌سازی نرم‌افزار

برای پیاده‌سازی نرم‌افزار از زبان برنامه نویسی C# با رویکرد شی‌گرا[۵۴] استفاده شد. همچنین مدل فرایند[۵۵] استفاده شده در این نرم‌افزار مدل فرایند آبشاری[۵۶] است که در ادامه به بررسی آن و همچنین نمودارهای لازم برای تحلیل نرم‌افزار پرداخته خواهد شد. علاوه بر آن نرم‌افزار کنونی دارای قابلیت توسعه‌پذیری است که روند اعمال و پیاده‌سازی آن در نرم‌افزار در ادامه توضیح داده خواهد شد. در نهایت واسط گرافیکی پیاده‌سازی شده برای نرم‌افزار مورد بررسی قرار می‌گیرد. برخی از کدهای پیاده‌سازی شده در نرم‌افزار در پیوست آمده است.

۱.۵ تحلیل و طراحی نرم‌افزار

مدل آبشاری یک مدل ترتیبی توسعه و تولید نرم‌افزار است و در آن مراحل تولید به شکل یک جریان مداوم متمایل به سمت پأیین می‌باشد. همانند یک آبشار که شامل فازهای تحلیل خواسته‌ها، طراحی، پیاده‌سازی[۵۷]، آزمون[۵۸]، یکپارچه سازی[۵۹] و دادن محصول به بازار می‌شود. مدیریت و مراحل تکمیل پروژه در این مدل فرایند به سادگی قابل پیاده‌سازی است. زیرا در مرحله اول که مرحله بررسی

---

[۵۴] Object oreinted

[۵۵] Process model

[۵۶] Waterfall

[۵۷] Implementation

[۵۸] Test

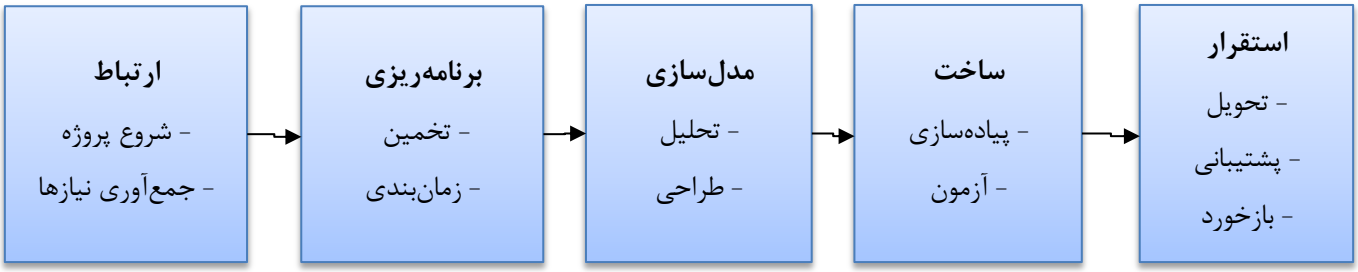
[۵۹] Integration

نیازمندی‌های پروژه می‌باشد، مشتری و تیم برنامه نویسی طی چند جلسه به بررسی نیازمندی‌ها و خواسته‌های پروژه می‌پردازند. پس از آن نوبت به مرحله طراحی می‌رسد، در مرحله‌ی طراحی افراد طرح کلی پروژه را می‌ریزند و جزییات پیاده‌سازی مشخص می‌شود. پس از مرحله‌ی طراحی تیم برنامه نویسی خود را برای پیاده‌سازی آماده می‌کند. در این مرحله همه قسمت‌های کد، پیاده‌سازی می‌شوند و در انتهای این مرحله، مرحله یکپارچه سازی را خواهیم داشت که یکی از مشکل‌ترین قسمت‌های انجام پروژه‌ها در این مرحله می‌باشد. زیرا تنوع و گستردگی کار کاملا در این مرحله نقش دارد، هر چه میزان گستردگی کار بالاتر باشد، سختی یکپارچه سازی نیز بیشتر خواهد بود.

در این پروژه با توجه به مشخص و ثابت بودن نیازهای نرم‌افزار در ابتدای تعریف پروژه، می‌توان از مدل فرآیند آبشاری استفاده کرد. علاوه بر آن این مدل فرآیند دارای مزیت‌هایی است که استفاده از آن را تایید می‌کند که در ادامه به مزیت‌های این مدل فرایند پرداخته می‌شود.

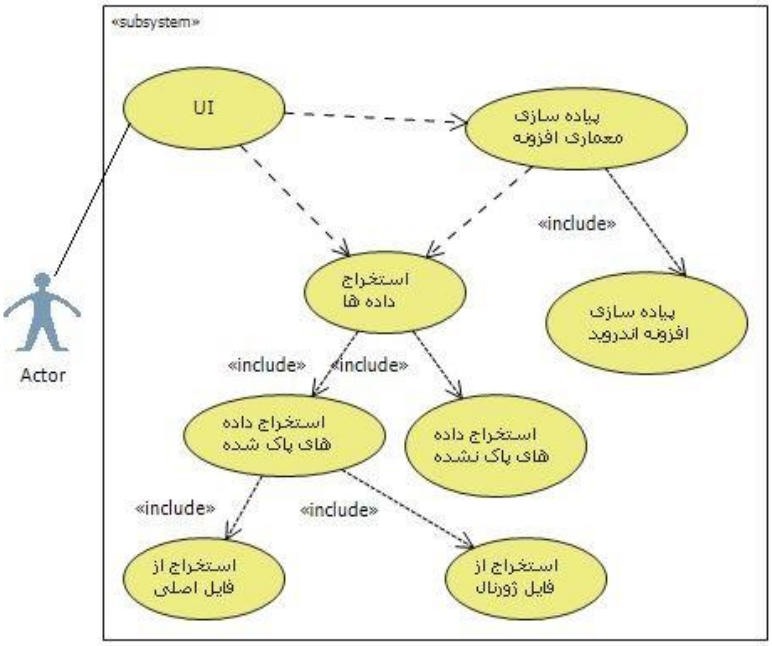
مزیت‌های مدل آبشاری به شرح زیر می‌باشد:

- فهم این مدل ساده‌تر است.
- از نظر تولید مستندات شرایط بهتر و آسان‌تری دارد.
- مراحل قابل کنترل و بررسی می‌باشند.

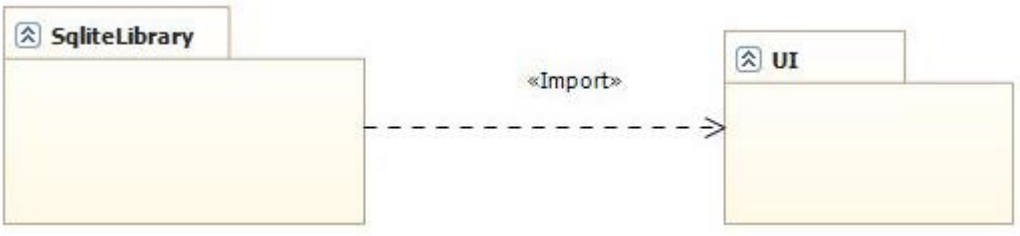


شکل ۲۲ فرآیند تولید نرم‌افزار به صورت آبشاری

در مرحله تحلیل نرم‌افزار ابتدا نمودار درخواست سیستم[۶۰] ایجاد می‌شود که در شکل ۲۳ قابل مشاهده است. در ادامه نمودارهای پکیج[۶۱] و کلاس[۶۲] در شکل‌های ۲۴، ۲۵ و ۲۶ آمده‌اند. گراف وابستگی کلاس‌ها هم در شکل ۲۷ آمده است.



شکل ۲۳ نمودار در خواست سیستم

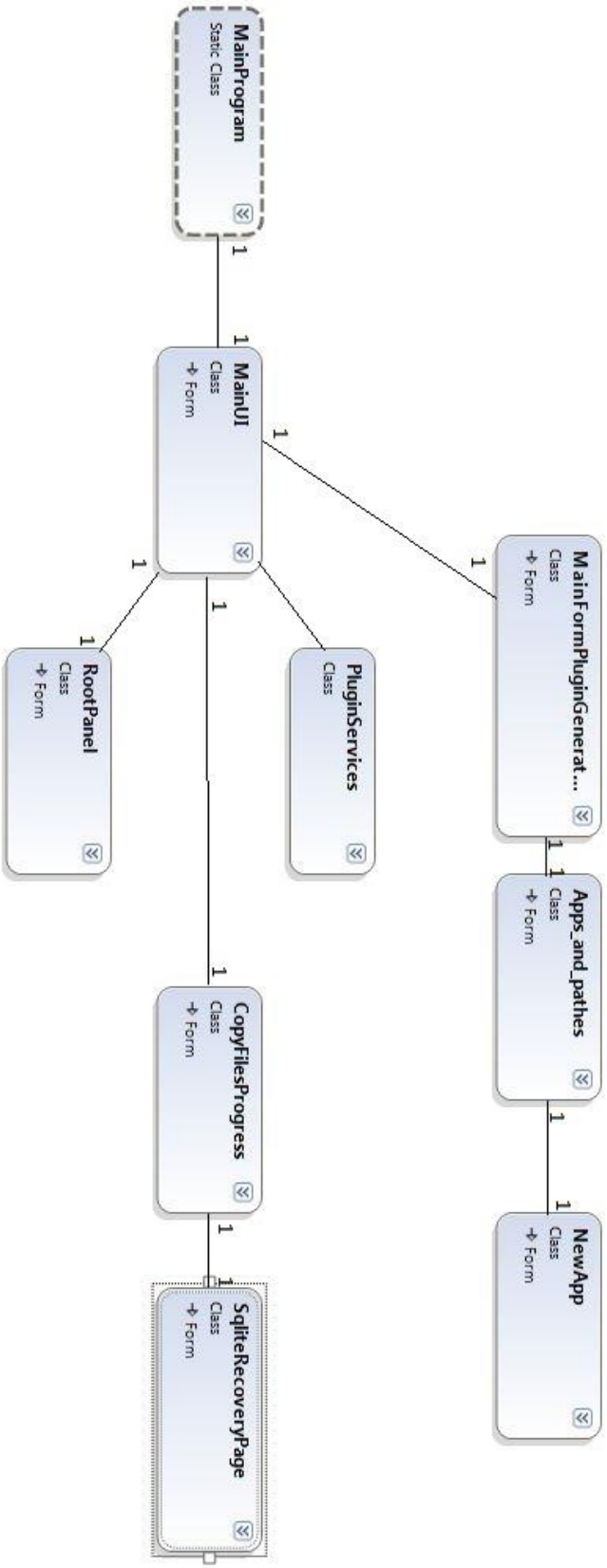


شکل ۲۴ نمودار پکیج

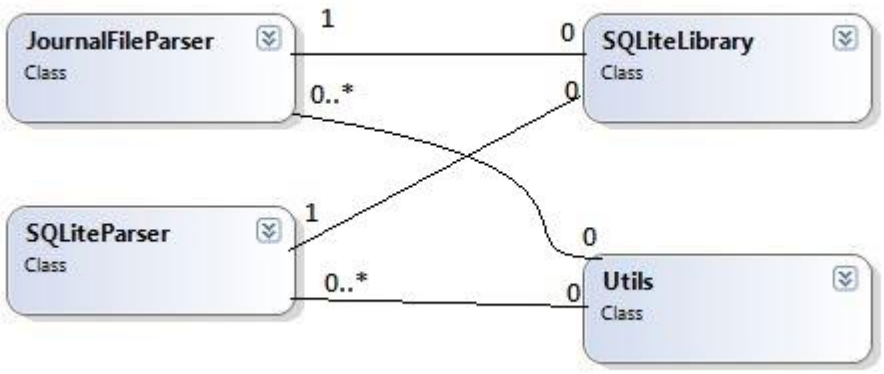
[۶۰] Usecase

[۶۱] Package

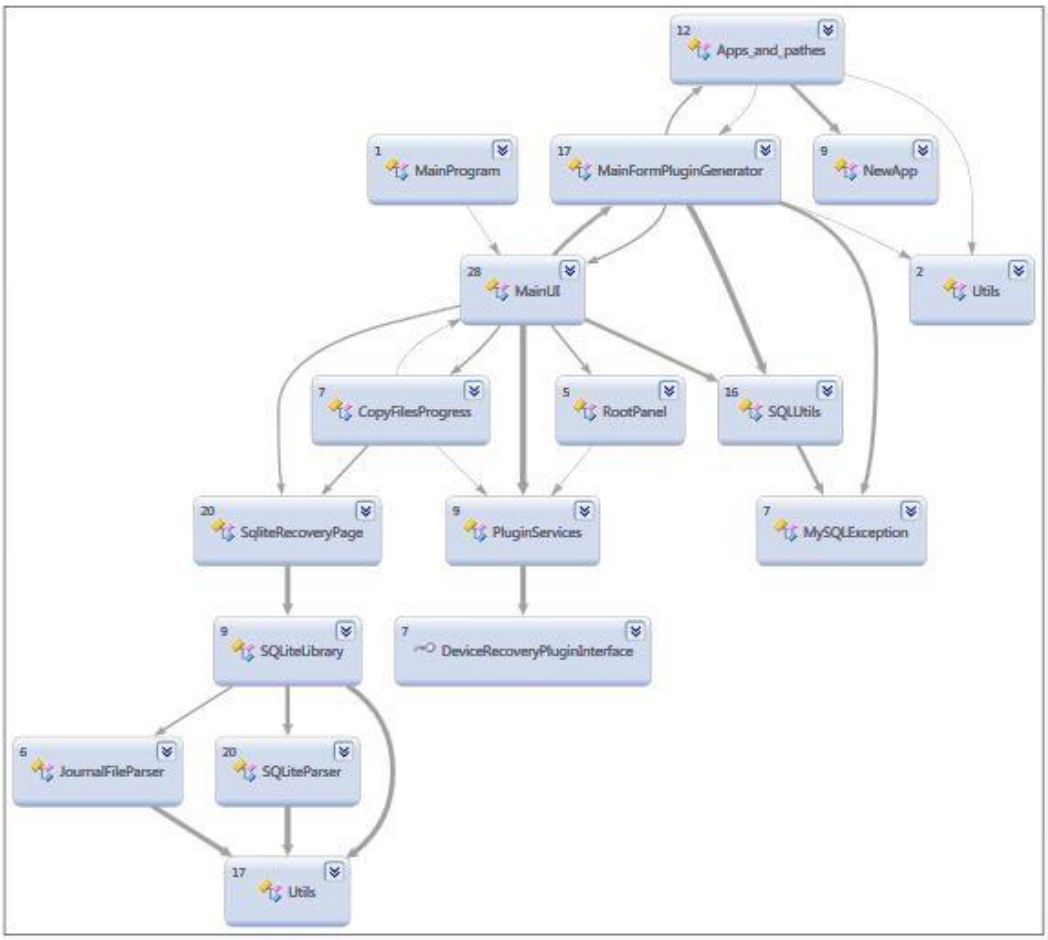
[۶۲] class



شکل ۲۵ نمودار class کتابخانه UI



شکل ۲۷ نمودار کلاس برای کتابخانه sqlitelibrary



شکل ۲۶ نمودار وابستگی کلاس‌ها

۲.۵ قابلیت توسعه‌پذیری

همان طور که پیش از این ذکر شد پایگاه‌داده Sqlite در پلتفرم‌های مختلف از جمله اندروید، iOS و ویندوز کاربرد دارد. برای اینکه نرم‌افزار حاضر بتواند از همه این پلتفرم‌ها، فایل‌های پایگاه‌داده را بگیرد و پردازش کند، قابلیت توسعه‌پذیری به سیستم اضافه شد. برای این منظور از طریق واسط کاربری که در بخش بعد به تفصیل توضیح داده خواهد شد نام و آدرس ذخیره‌سازی پایگاه‌داده‌های هر نرم‌افزار، آدرس فایل dll (شامل دستورات لازم برای اتصال، کپی کردن و غیره به زبان C# است. این دستورات به واسطه یک اینترفیس[۶۳] باید پیاده‌سازی شوند که کد آن در شکل ۲۸ آمده است. سرانجام این کدها کامپایل شده و فایل dll تولید می‌شود.) و نوع سیستم عامل، دریافت شده و یک افزونه تولید می‌شود. این افزونه از طریق بارگذاری فایل‌های اسمبلی در C#، فایل dll را در نرم‌افزار بارگذاری کرده و از تابع‌های موجود در اینترفیس استفاده می‌کند. برای کنترل اینکه فایل بارگذاری‌شده معتبراست، نوع فایل بارگذاری شده

```
using System;
using System.Collections.Generic;
using System.Linq;
using System.Text;

namespace DevicePluginInterface
{
    public interface DeviceRecoveryPluginInterface
    {
        void copyAppDataBaseFromDevice(string key,string path, string distination);

        bool isDeviceRoot();

        bool rootDevice();

        bool unRootDevice();

        bool isDeviceConnected();

        void refreshDeviceList();

        bool installApp(string path);
    }
}
```

شکل ۲۸ نمایی از کد اینترفیس و تابع‌هایی که برای هر افزونه باید پیاده‌سازی شوند.

[۶۳] Interface

باید از نوع اینترفیس باشد یعنی اینترفیس باید در آن پیاده‌سازی شده باشد. در این پروژه یک افزونه به منظور اتصال به دستگاه‌های اندرویدی پیاده‌سازی شده است. برای پیاده‌سازی این افزونه از کتابخانه Android Lib استفاده شد که دستورات ADB Shell در آن پیاده‌سازی شده‌اند.

۳.۵ واسط کاربری گرافیکی[۶۴]

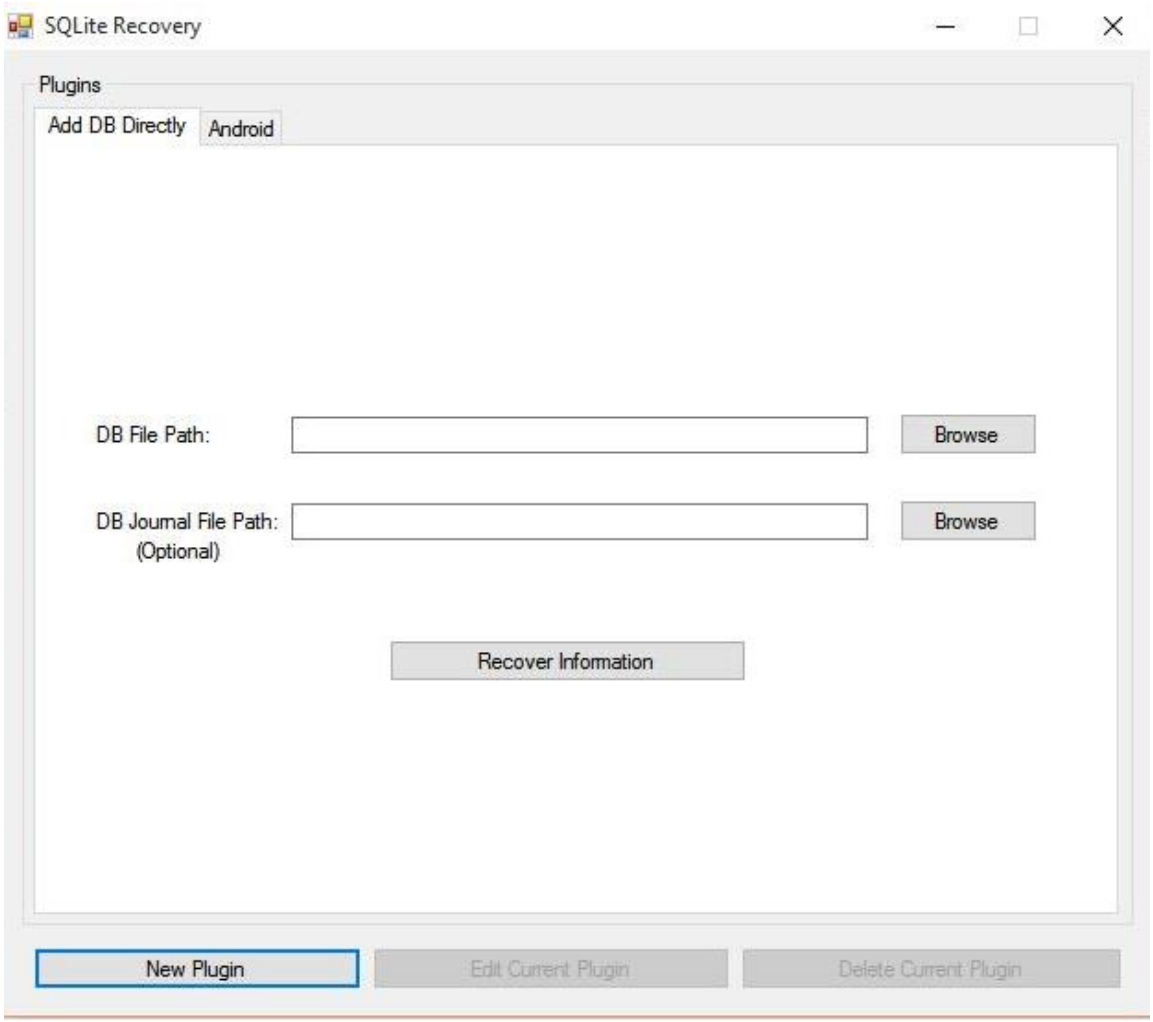
اهمیت ظاهر برنامه و صفحاتی که کاربر توسط آن ها با سیستم در تعامل است، بر کسی پوشیده نیست. در پروژه حاضر به دلیل وجود قابلیت توسعه‌پذیری این مورد اهمیت بیشتری پیدا می‌کند چون واسط کاربری باید به گونه‌ای باشد که کاربر به راحتی بتواند افزونه‌های مورد نیاز خود را تولید کرده و به سیستم اضافه کند. علاوه بر آن واسط کاربری باید امکانات لازم برای دسترسی به اجزای مختلف پایگاه‌داده‌ها را فراهم آورد که این موارد از طریق اضافه کردن Tab، ComboBox و DataGridView فراهم آمده است.

یک واسط کاربری خوب دارای ویژگی‌هایی است که در زیر به برخی از آنها اشاره شده‌است:

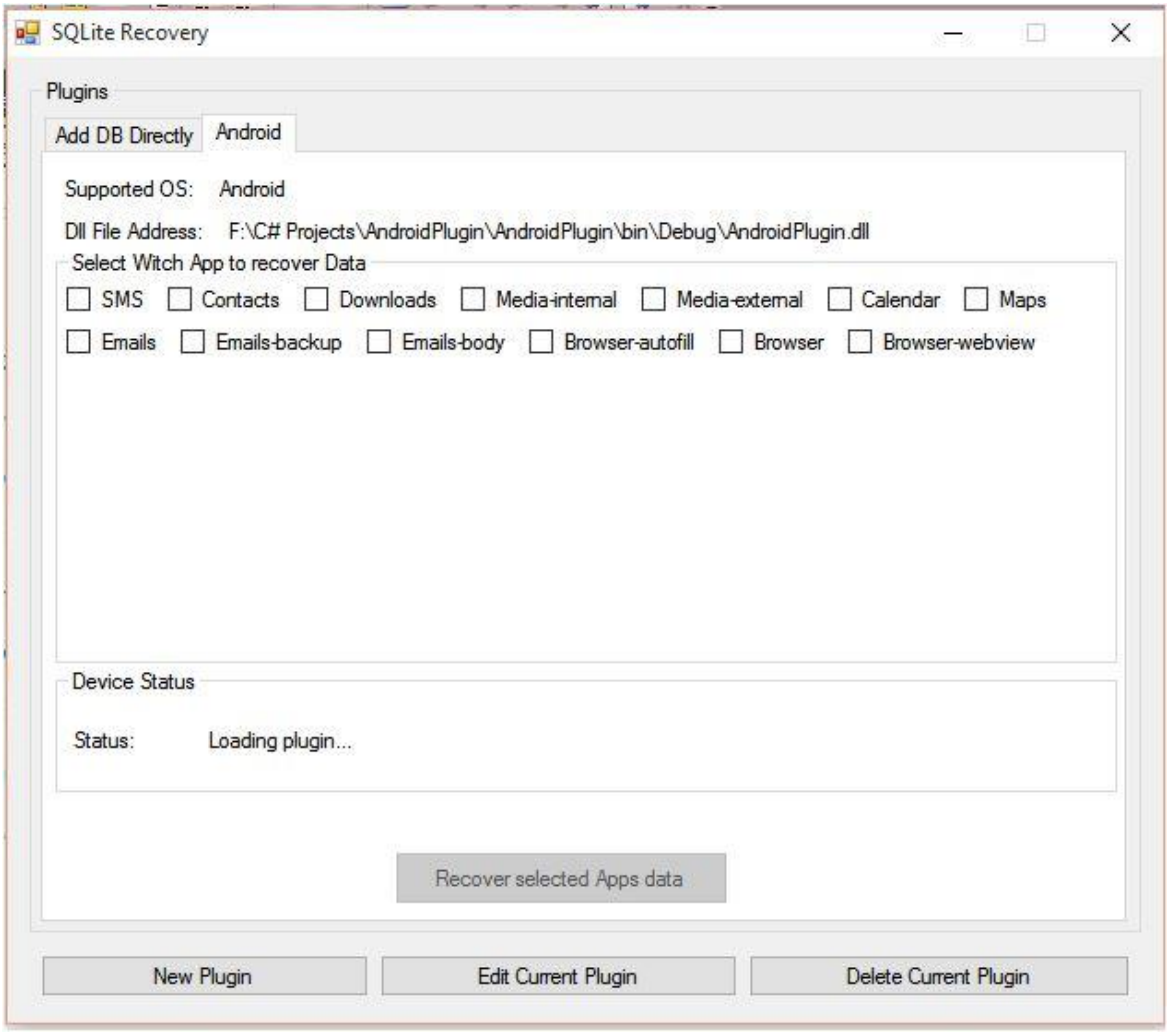
- گزینه‌ها و دکمه های موجود در صفحه باید همگون و با سبک یکسان باشند.
- در هنگام تغییر وضعیت برنامه، باید ظاهر نیز متناسب با آن تغییر یابد. یعنی برنامه متناسب با هر فعالیت، بازخورد مناسبی داشته باشد.
- هر گزینه باید کاملا واضح و دارای معنای خاص باشد.
- برای همگی فعالیت‌ها، حالت های پیش فرض در نظر گرفته شود.
- کاربر نیازی به آموزش برای یادگیری کار با رابط کاربری نداشته باشد یا حداقل باشد.
- اجزائی که با یکدیگر مرتبط هستند، در یک گروه‌بندی خاص باشند
- برای حذف یا پاک کردن اطلاعات مهم، تأیید مجدد کاربر دریافت شود.
- امکان تغییر ابعاد صفحه برای کاربر وجود داشته باشد و ضمناً با تغییر ابعاد پنجره برنامه، چینش اجزا در صفحه منظم باقی بماند

[۶۴] Graphical user interface(GUI)

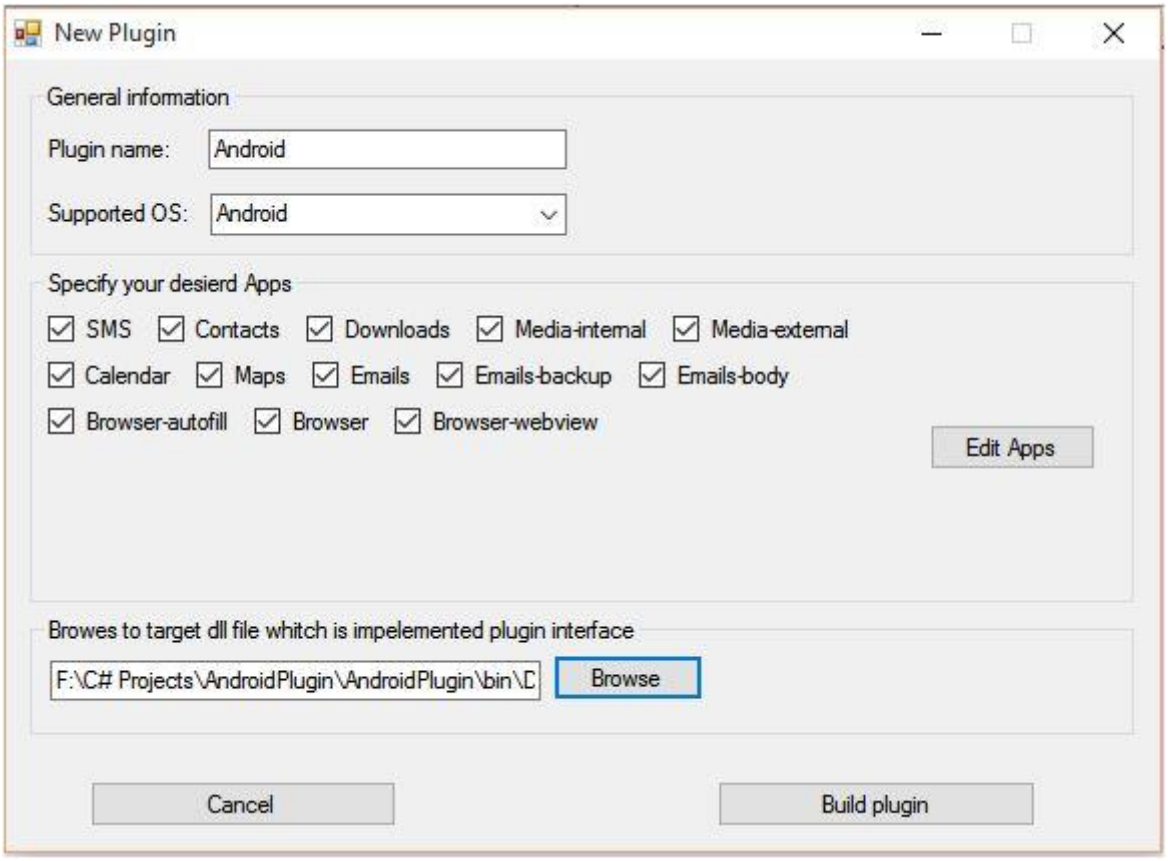
در این پروژه سعی شده‌است موارد بالا رعایت شود. در ادامه نمایی از واسط‌های کاربری پیاده‌سازی شده آمده است.



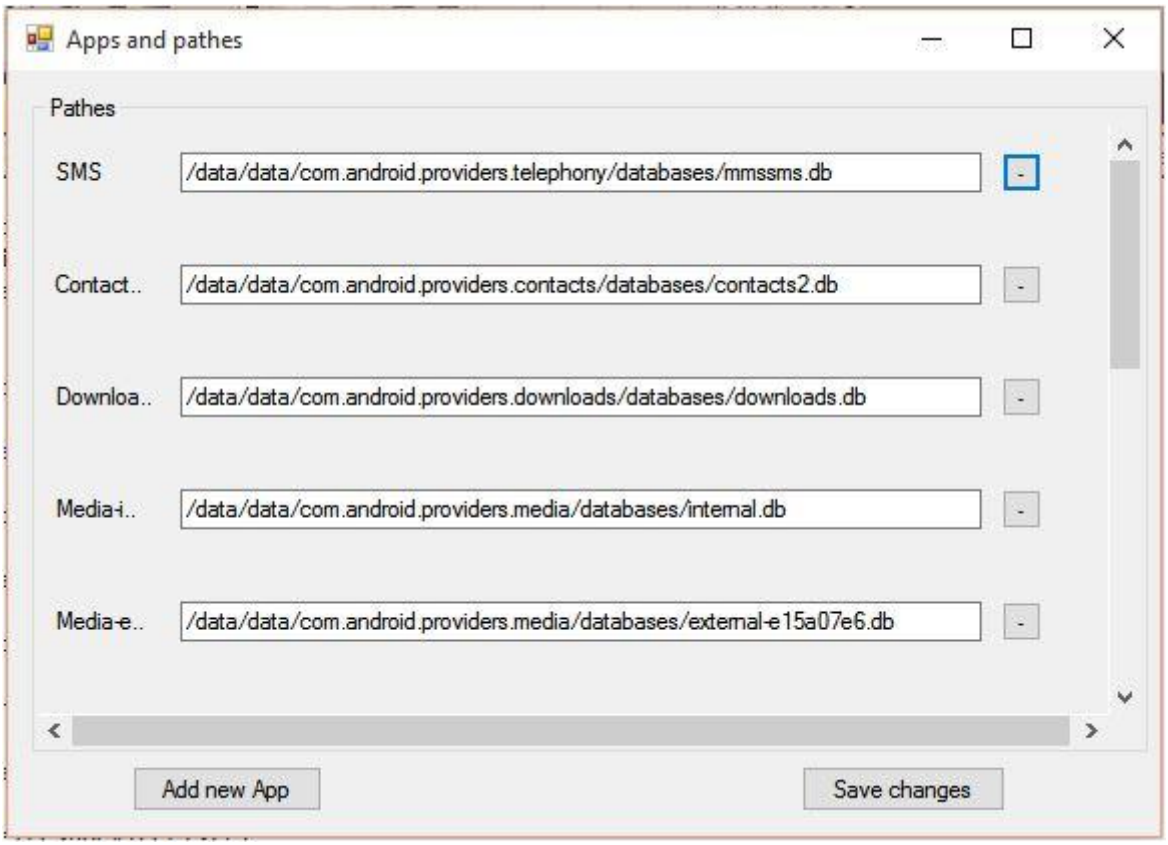
شکل ۲۹ صفحه اصلی برنامه، پردازش پایگاه‌داده‌های موجود در رایانه



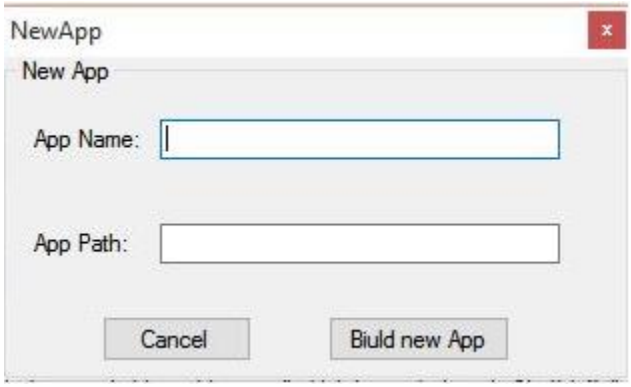
شکل ۳۰ نمایی از یک افزونه



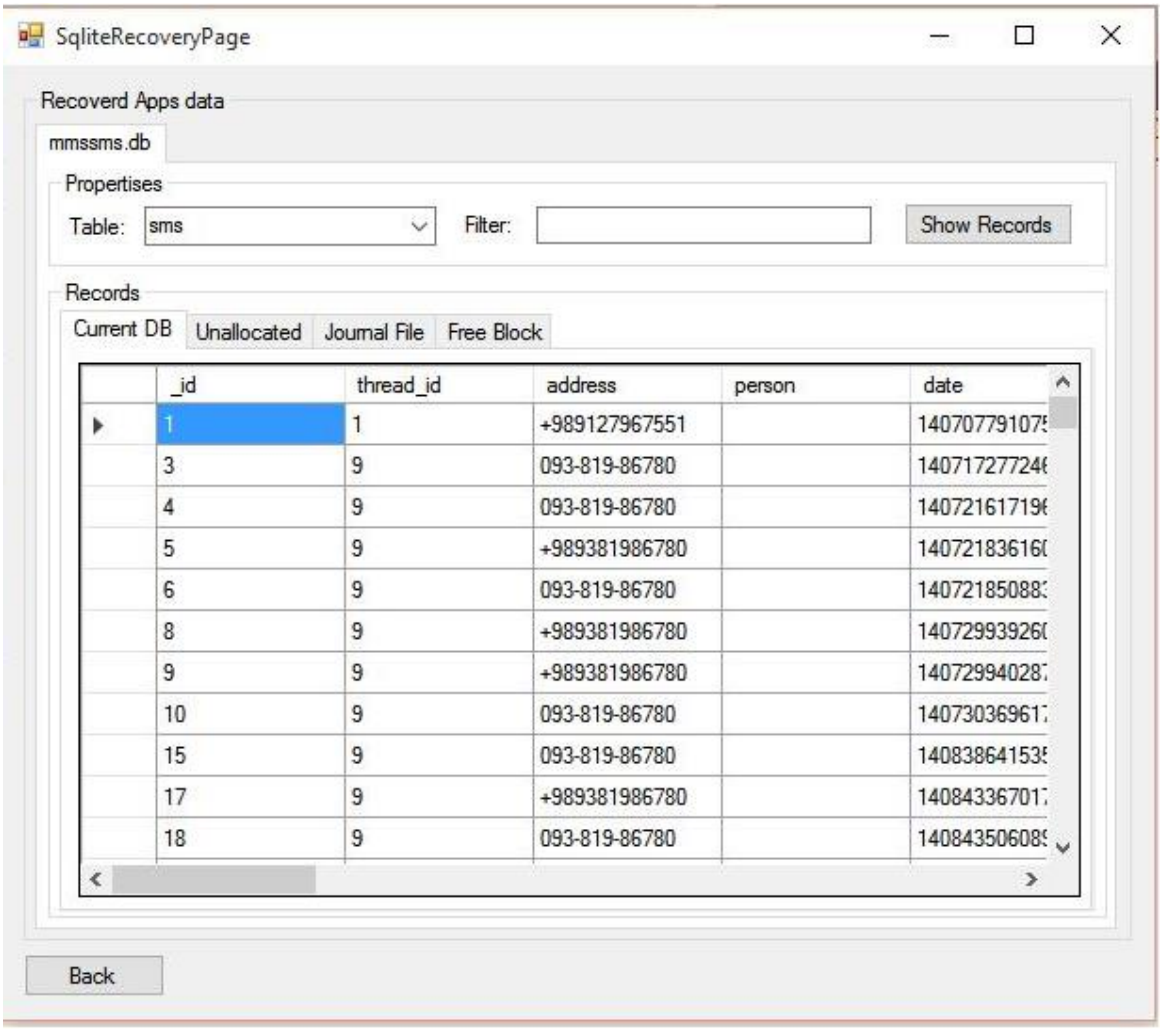
شکل ۳۱ ایجاد افزونه جدید



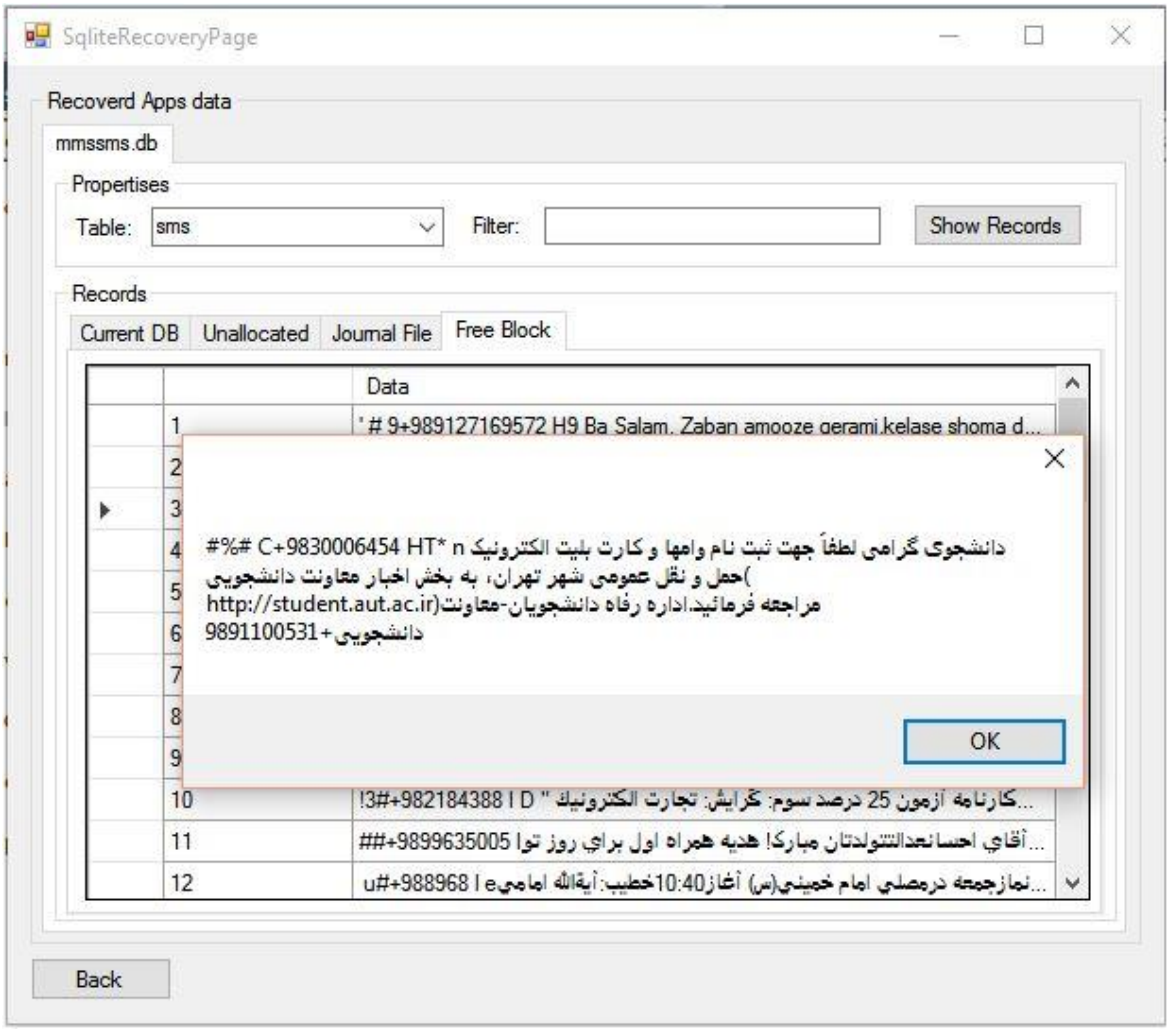
شکل ۳۲ حذف یا اضافه کردن نام و آدرس نرم‌افزار به افزونه



شکل ۳۳ اضافه کردن نام و آدرس نرم‌افزار



شکل ۳۴ صفحه داده‌های پایگاه‌داده‌ها



شکل ۳۵ نمایی از داده‌های پاک‌شده

۴.۵ جمع‌بندی

در این فصل روند پیاده‌سازی ابزار مورد بررسی قرار گرفت. ابتدا نحوه تحلیل و طراحی نرم‌افزار بیان شد همچنین مدل فرایند آبشاری که برای پیاده‌سازی نرم‌افزار از آن استفاده شد مورد بررسی قرار گرفت در ادامه نحوه تحلیل و نمودارهای تحلیل مورد استفاده، بررسی شدند. بعد از آن قابلیت توسعه‌پـذیری و نحوه اعمال آن در برنامه بیان شد و در پایان نکته‌های لازم برای پیاده‌سازی واسط گرافیکی بیان شدند و نمایی از واسط گرافیکی ابزار نشان داده شد.

۶

فصل ششم

جمع‌بندی و کارهای آینده

هدف از این پروژه پیاده‌سازی ابزاری با قابلیت توسعه‌پذیری برای استخراج داده‌های پاک‌شده از گوشی‌های هوشمند بود. همان طور که پیش از این ذکر شد گوشی‌های هوشمند به دلیل استفاده آسان و گستره قیمتی متنوع در میان مردم به طور فراگیر گسترش یافته است. از طرف دیگر این ابزارها همانند رایانه‌ها مبتنی بر سیستم عامل بوده و نرم‌افزارهای گوناگونی بر روی این ابزارها اجرا می‌شوند. این نرم‌افزارها هر کدام به تنهایی اطلاعات زیادی در مورد دارنده گوشی در خود ذخیره می‌کنند. به طور مثال نرم‌افزار پیام‌رسان، پیام‌های ارسالی بین دارنده گوشی و افراد در ارتباط با او را در خود ذخیره می‌کند.

امروزه از این نوع اطلاعات برای انجام تحقیقات و اثبات جرم توسط مراجع قضایی و پلیس استفاده می‌شود. این موضوع اهمیت داده‌های پاک‌شده را دوچندان می کند چرا که امکان دارد این داده‌ها توسط دارنده گوشی عمدا پاک شده باشد.

از آنجایی که گوشی‌های مختلف دارای سیستم عامل‌های متفاوتی هستند ارتباط با گوشی‌های مختلف متفاوت خواهد بود که برای حل این مشکل قابلیت توسعه پذیری به سیستم اضافه شد. تا به وسیله آن بتوان از گوشی‌های مختلف پشتیبانی کرد. در این پروژه تمرکز بروی گوشی‌های مبتی بر اندروید و استخراج داده از پایگاه‌داده Sqlite بود.

در فصول گذشته روند تعریف مسئله و چگونگی پیاده‌سازی نرم‌افزار و روند آن مورد بررسی قرار گرفت که در ادامه به بررسی هر یک از این فصول می پردازیم.

در فصل اول به طور اجمالی به تعریف مسئله و اهمیت موضوع پرداخته شد. در این فصل علت اهمیت اطلاعات ذخیره شده در گوشی‌های هوشمند و دلیل اهمیت داده‌های پاک‌شده مورد بررسی قرار

گرفت. همچنین بیان شد، برای این که ابزار پیاده‌سازی شده، گوشی‌ها و سیستم عامل‌های گوناگون را پشتیبانی کند نیاز است تا قابلیت توسعه‌پذیری به سیستم اضافه شود. همچنین محدوده‌ها و ساختار پروژه به طور کلی عنوان شد و بیا شد که تاکید پروژه کنونی بر روی سیستم عامل اندروید و بازیابی اطلاعات از پایگاه‌داده Sqlite است.

درفصل دوم به طور اجمالی سیستم عامل اندروید مورد بررسی قرار گرفت. ابتدا تاریخچه‌ای کوتاه و همچنین نسخه‌های مختلف آن بررسی شد. در ادامه معماری این سیستم عامل و نحوه پیاده‌سازی و اجرای نرم‌افزارهای مختلف تحت اندروید بررسی شد. سپس مدل امنیتی اندروید و نحوه پنهان سازی اطلاعات هر نرم‌افزار از دیگری مورد بررسی قرار گرفت. در پایان نحوه اتصال گوشی به رایانه و استفاده از ابزارهای ADB و Kingo Root بیان شدند.

در فصل سوم ساختار فایل‌های پایگاه‌داده Sqlite مورد بررسی قرار گرفت. همان طور که بیان شد این پایگاه‌داده از دو فایل اصلی و فایل ژورنال تشکیل شده‌اند. در این فصل ساختار این دو فایل و نحوه ذخیره‌سازی و پاک‌شدن رکوردها از پایگاه داده بیان شدند.

در فصل چهارم الگوریتم‌ها و روش‌های بازیابی اطلاعات از پایگاه‌داده Sqlite مورد بررسی قرار گرفت. همان‌طور که در این فصل بیان شد دو روش برای بازیابی اطلاعات از فایل اصلی پایگاه‌داده و یک روش برای بازیابی به کمک فایل ژورنال پیشنهاد شد. همچنین نحوه استفاده از ابزار sqldiff برای مقایسه دو پایگاه‌داده Sqlite مورد بررسی قرار گرفت.

در فصل پنجم روند پیاده‌سازی ابزار مورد بررسی قرار گرفت. ابتدا نحوه تحلیل و طراحی نرم‌افزار بیان شد همچنین مدل فرایند آبشاری که برای پیاده‌سازی نرم‌افزار از آن استفاده شد مورد بررسی قرار گرفت در ادامه نحوه تحلیل و نمودارهای تحلیل مورد استفاده، بررسی شدند. بعد از آن قابلیت توسعه‌پذیری و نحوه اعمال آن در برنامه بیان شد و در پایان نکته‌های لازم برای پیاده‌سازی واسط گرافیکی بیان شدند و نمایی از واسط گرافیکی ابزار نشان داده شد.

آنچه که در این پروژه مورد بررسی قرار گرفت بازیابی بخش کوچکی از داده‌های ذخیره شده در ابزارهای هوشمند اندرویدی بود. علاوه بر آن ابزارهای هوشمند اندرویدی دارای طیف گسترده‌ای از نظر نسخه سیستم عامل استفاده شده و پلتفرم سخت‌افزاری هستند، که نحوه دسترسی به فایل‌های

پایگاه‌داده در آنها ممکن است اندکی متفاوت باشد، که این مورد نیز باید مورد بررسی قرار گیرد. علاوه بر سیستم عامل اندروید، سیستم عامل iOS و ویندوز نیز باید مورد بررسی قرار گیرند. علاوه بر داده‌های ذخیره شده در Sqlite داده‌های دیگر از جمله عکس‌های ذخیره شده نیز می‌توانند موضوع تحقیق قرار گیرند. البته ذکر این نکته لازم است که پیاده‌سازی ویژگی توسعه‌پذیری این امکان را می‌دهد که پیاده‌سازی موارد بالا و اضافه کردن آنها به نرم‌افزار کنونی راحت‌تر صورت پذیرد.

منابع و مراجع

[1] Paraben.com, '*Paraben - Device Seizure: DS 7 Mobile Forensic Software*', 2015. [Online]. Available: https://www.paraben.com/device-seizure.html. [Accessed: 21- Oct- 2015].

[2] A. Hoog and J. McCash, '*Android Forensics*'. Waltham, MA: Syngress, 2011.

[3] Developer.android.com, '*Download Android Studio and SDK Tools | Android Developers*', 2015. [Online]. Available: https://developer.android.com/sdk/index.html. [Accessed: 02- Oct- 2015].

[4] Sqlite.org, '*File Format For SQLite Databases*', 2015. [Online]. Available: http://sqlite.org/fileformat2.html. [Accessed: 02- Oct- 2015].

[5] Ray.bsdart.org, '*SQLite Database File Format*', 2015. [Online]. Available: http://ray.bsdart.org/man/sqlite/fileformat.html. [Accessed: 22- Oct- 2015].

پیوست

در این قسمت کدهای پیاده‌سازی شده برای هر کتابخانه آمده‌اند. کلاس پیاده‌سازی شـده بـرای بازیابی اطلاعات از فایل اصلی در زیر آمده است.

```
class SQLiteParser
{
private const int
internalNodeTypeValue = 5;
private const int
leafNodeTypeValue = 13;
private int pageSize;
private int rootPageNum;
private const int
leafPageHeaderLength = 8;
private const int
internalPageHeaderLength = 12;
private const int
pageSizeOffsetValue = 16;
private const int
pageSizeLengthValue = 2;
private const string
headerString="SQLite format 3\0";
private byte[] currentPage;
private byte[]
headerBytesOfSQLiteFile;
private const int dbHeaderSize =
100;
private string dbFilePath;
private const int
internalPageCellsPionerLength = 4;
private ArrayList tableInfo;
private ArrayList
UnallocatedSpaceDeletedRecords {
get; set; }
private Dictionary<int, ArrayList>
sqliteTypes;

private ArrayList records = new
ArrayList();
internal string dbCopyFilePath;
```

```
internal SQLiteParser(string
dbFilePath)
{
this.dbFilePath = dbFilePath;
this.UnallocatedSpaceDeletedRecord
s = new ArrayList();

init();

}

private void init()
{
sqliteTypes = new Dictionary<int,
ArrayList>();
sqliteTypes.Add(0, new ArrayList()
{ "NULL", "0" });
sqliteTypes.Add(1, new ArrayList()
{ "INTEGER_1", "1" });
sqliteTypes.Add(2, new ArrayList()
{ "INTEGER_2", "2" });
sqliteTypes.Add(3, new ArrayList()
{ "INTEGER_3", "3" });
sqliteTypes.Add(4, new ArrayList()
{ "INTEGER_4", "4" });
sqliteTypes.Add(5, new ArrayList()
{ "INTEGER_6", "6" });
sqliteTypes.Add(6, new ArrayList()
{ "INTEGER_8", "8" });
sqliteTypes.Add(7, new ArrayList()
{ "FLOAT", "8" });
sqliteTypes.Add(8, new ArrayList()
{ "INTEGER_0", "0" });
sqliteTypes.Add(9, new ArrayList()
{ "INTEGER_0", "0" });
sqliteTypes.Add(12, new
ArrayList() { "BLOB", "*" });
sqliteTypes.Add(13, new
ArrayList() { "STRING", "*" });

headerBytesOfSQLiteFile =
Utils.ReadingFromFile(dbFilePath,
0, dbHeaderSize);
byte[] currentHeaderString = new
byte[16];
Array.Copy(headerBytesOfSQLiteFile
, currentHeaderString, 16);
string cur =
System.Text.Encoding.ASCII.GetStri
ng(currentHeaderString);

if (headerString.Equals(cur))
```

```
getDeletedRecordsFromUnallocatedSp
aceAndFreeBlocks(numOfCells,
cellsOffset, currentPageNum);

return;
}
else if (currentPage[0] ==
internalNodeTypeValue)
{
int[] childPtr =
getChildsPtr(numOfCells, ref
cellsOffset);
getDeletedRecordsFromUnallocatedSp
aceAndFreeBlocks(numOfCells,
cellsOffset, currentPageNum);

foreach (int ptr in childPtr)
{
    BTreeTraversal(ptr);
}
}
}

private int[] getChildsPtr(int
numOfCells, ref int cellsOffset)
{
int[] childPtr = new
int[numOfCells + 1];
int offset = 12;
int pointer = cellsOffset;
for (int i = 0; i <
childPtr.Length - 1; i++)
{
cellsOffset =
BitConverter.ToInt16(new byte[] {
currentPage[offset + 1],
currentPage[offset] }, 0);
offset = offset + 2;
childPtr[i] =
BitConverter.ToInt32(new byte[] {
currentPage[cellsOffset + 3],
currentPage[cellsOffset + 2],
currentPage[cellsOffset + 1],
currentPage[cellsOffset] }, 0);
}
childPtr[childPtr.Length - 1] =
BitConverter.ToInt32(new byte[] {
currentPage[11], currentPage[10],
currentPage[9], currentPage[8] },
0);
return childPtr;
}
{
byte[] result =
Utils.ReadingFromFile(dbFilePath,
pageSizeOffsetValue,
pageSizeLengthValue);
Array.Reverse(result);
pageSize =
BitConverter.ToInt16(result, 0);

dbCopyFilePath = dbFilePath +
"_c";

System.IO.File.Copy(dbFilePath,
dbCopyFilePath,true);

tableInfo =
Utils.getAllTablesInfo(dbCopyFileP
ath);
}
else
{
throw new
FileNotFoundException("Type of
given file is not SQLite!!",
dbFilePath);
}

}

private void BTreeTraversal(int
currentPageNum)
{
int currentPageOffset = 0;
if (currentPageNum != 0)
currentPageOffset =
(currentPageNum - 1) * pageSize;
currentPage =
Utils.ReadingFromFile(dbFilePath,
currentPageOffset, pageSize);
int numOfCells =
BitConverter.ToInt16(new byte[] {
currentPage[4], currentPage[3] },
0);
int cellsOffset =
BitConverter.ToInt16(new byte[] {
currentPage[6], currentPage[5] },
0);

if (currentPage[0] ==
leafNodeTypeValue)
{
```

```
}
else if (nextFreeBlockOffset == 0)
{
getDataFromFreeBlock(currentFreeBl
ockOffset, currentFreeBlockSize,
pageNum);
return;
}
}

private void
getDataFromUnallocatedSpace(int
unallocatedSpaceOffset, int
cellsOffset, int pageNum)
{
string data =
Encoding.UTF8.GetString(currentPag
e, unallocatedSpaceOffset,
cellsOffset -
unallocatedSpaceOffset);

data = data.Replace("\0", "");
foreach (char c in data)
{
if (!((c >= 0x0600 && c <= 0x06FF)
|| (c >= 0x0021 && c <= 0x007E) ||
c == 0xFB8A || c == 0x067E || c ==
0x0686 || c == 0x06AF))
{
    data = data.Replace(c, ' ');

}

}
data = data.Replace("  ",
string.Empty);

if (!String.IsNullOrEmpty(data))
UnallocatedSpaceDeletedRecords.Add
(new string[] { pageNum + "",
"UNALLOCATED", data });
}

private void
getDataFromFreeBlock(int
currentFreeBlockOffset, int
currentFreeBlockSize, int pageNum)
{
string data =
Encoding.UTF8.GetString(currentPag
e, currentFreeBlockOffset + 4,
currentFreeBlockSize - 4);

private void
getDeletedRecordsFromUnallocatedSp
aceAndFreeBlocks(int numOfCells,
int cellsOffset, int pageNum)
{
getAllFreeBlockListData(BitConvert
er.ToInt16(new byte[] {
currentPage[2], currentPage[1] },
0), pageNum);
getDataFromUnallocatedSpace(numOfC
ells * 2 + leafPageHeaderLength,
cellsOffset, pageNum);
}

private void
getAllFreeBlockListData(int
currentFreeBlockOffset, int
pageNum)
{
int currentFreeBlockSize =
BitConverter.ToInt16(new byte[] {
currentPage[currentFreeBlockOffset
+ 3],
currentPage[currentFreeBlockOffset
+ 2] }, 0);
int nextFreeBlockOffset =
BitConverter.ToInt16(new byte[] {
currentPage[currentFreeBlockOffset
+ 1],
currentPage[currentFreeBlockOffset
] }, 0);

if (currentFreeBlockOffset == 0)
{
nextFreeBlockOffset =
BitConverter.ToInt16(new byte[] {
currentPage[2], currentPage[1] },
0);
currentFreeBlockSize = 0;
if (nextFreeBlockOffset == 0)
    return;
}

if (nextFreeBlockOffset != 0)
{
getDataFromFreeBlock(currentFreeBl
ockOffset, currentFreeBlockSize,
pageNum);

getAllFreeBlockListData(nextFreeBl
ockOffset, pageNum);
```

```
int firstTrunkPageNum =
BitConverter.ToInt32(new byte[] {
headerBytesOfSQLiteFile[35],
headerBytesOfSQLiteFile[34],
headerBytesOfSQLiteFile[33],
headerBytesOfSQLiteFile[32] }, 0);
if (firstTrunkPageNum != 0)
{
getDeletedPagesFromFreeList(firstT
runkPageNum);
}
return records;
}

private void
getDeletedPagesFromFreeList(int
trunkPageNum)
{
int offset=(trunkPageNum-
1)*pageSize;
currentPage =
Utils.ReadingFromFile(dbFilePath,
offset, pageSize);
int nextTrunkPageNum =
BitConverter.ToInt32(new byte[] {
currentPage[3], currentPage[2],
currentPage[1], currentPage[0] },
0);
int numOfLeafPagesHear =
BitConverter.ToInt32(new byte[] {
currentPage[7], currentPage[6],
currentPage[5], currentPage[4] },
0);
offset = 8;
int [] freepages=new
int[numOfLeafPagesHear];
for (int i = 0; i <
numOfLeafPagesHear; i++)
{
freepages[i] =
BitConverter.ToInt32(new byte[] {
currentPage[offset + 3 + i * 4],
currentPage[offset + 2 + i * 4],
currentPage[offset + 1 + i * 4],
currentPage[offset + i * 4] }, 0);
offset = offset + 4;
}

foreach (int pageNum in freepages)
{
if (pageNum != 0)
data = data.Replace("\0", "");
foreach (char c in data)
{
if (!((c >= 0x0600 && c <= 0x06FF)
|| (c >= 0x0021 && c <= 0x007E) ||
c == 0xFB8A || c == 0x067E || c ==
0x0686 || c == 0x06AF))
{
    data = data.Replace(c, ' ');

}

}
data = data.Replace("  ",
string.Empty);

if (!String.IsNullOrEmpty(data))
UnallocatedSpaceDeletedRecords.Add
(new string[] { pageNum + "",
"FREE BLOCK", data });
}

internal
Dictionary<string,ArrayList>
UnAllocatedSpacesParser()
{
Dictionary<string, ArrayList> res
= new Dictionary<string,
ArrayList>();

foreach (string[] item in
tableInfo)
{
BTreeTraversal(Convert.ToInt32(ite
m[1]));
if(UnallocatedSpaceDeletedRecords.
Count>0)

res.Add(item[0],(ArrayList)Unalloc
atedSpaceDeletedRecords.Clone());
UnallocatedSpaceDeletedRecords.Cle
ar();

}

return res;
}

internal ArrayList
FreeListPagesParser()
{
records.Clear();
```

```
{
if (ptr == 801)

    Debug.Write("");
readDbRecordFromCell(ptr);
}

}
/// <summary>
/// exteract db records from given
cell offset.
/// </summary>
/// <param name="ptr"></param>
private void
readDbRecordFromCell(int ptr)
{
if (ptr == 247)
Debug.Write("");
byte[] buffer=new byte[9];
bool isOverflowPage=false;
int nextOverflowPage=0;
try
{
Array.Copy(currentPage, ptr,
buffer, 0, 9);
}
catch (ArgumentException ex)
{
Array.Copy(currentPage, ptr,
buffer, 0, currentPage.Length -
ptr);
}
long recordSizeValue=0;
int
recordSizeArrayLength=Utils.varInt
2Int(buffer, ref recordSizeValue);
ptr = ptr + recordSizeArrayLength;
try
{
Array.Copy(currentPage, ptr,
buffer, 0, 9);
}
catch (ArgumentException ex)
{
Array.Copy(currentPage, ptr,
buffer, 0, currentPage.Length-
ptr);
}
long keyValueField = 0;
int keyValueFieldLength =
Utils.varInt2Int(buffer, ref
keyValueField);
getDeletedRecordsFromDeletedPage(p
ageNum);
}

if (nextTrunkPageNum != 0)
getDeletedPagesFromFreeList(nextTr
unkPageNum);
else return;

}

private void
getDeletedRecordsFromDeletedPage(i
nt deletedLeafPageNum)
{
BTreeTraversal(deletedLeafPageNum)
;
}

private void
getRecordsFromLeafPagesInTableBTre
e(int currentPageNum)
{
if (currentPageNum == 530)
Debug.Write("");
int currentPageOffset = 0;
if (currentPageNum != 0)
currentPageOffset =
(currentPageNum - 1) * pageSize;
currentPage =
Utils.ReadingFromFile(dbFilePath,
currentPageOffset, pageSize);
int numOfCells =
BitConverter.ToInt16(new byte[] {
currentPage[4], currentPage[3] },
0);

int []cellsOffset=new
int[numOfCells];
int offset = 8;
for (int i = 0; i <
cellsOffset.Length; i++)
{
cellsOffset[i] =
BitConverter.ToInt16(new byte[] {
currentPage[offset + 1],
currentPage[offset] }, 0);
offset = offset + 2;
}

foreach (int ptr in cellsOffset)
```

```
    ArrayList item =
extractCurrentColLength(ref ptr,
buffer,0,ref recordHeaderSize);
    schema.Add(i, item);
}
}
else
{
long recordHeaderSize = 0;
Array.Copy(currentPage, ptr,
buffer, 0, 9);
int index =
Utils.varInt2Int(buffer, ref
recordHeaderSize);
ptr = ptr + index;
recordHeaderSize =
recordHeaderSize - index;

int colNum = 0;
for (; ptr <
nextOverFlowPageNumFieldOffset &&
recordHeaderSize != 0; colNum++)
{
    if (ptr == 260)
        Debug.Write("");
    ArrayList item=
extractCurrentColLength(ref ptr,
buffer,
nextOverFlowPageNumFieldOffset,ref
recordHeaderSize);
    schema.Add(colNum, item);
}
if (ptr ==
nextOverFlowPageNumFieldOffset)
{
    ptr = ptr -
(int)nextOverFlowPageNumFieldOffse
t + 4;
    nextOverflowPage =
BitConverter.ToInt32(new byte[] {
currentPage[nextOverFlowPageNumFie
ldOffset + 3],
currentPage[nextOverFlowPageNumFie
ldOffset + 2],
currentPage[nextOverFlowPageNumFie
ldOffset + 1],
currentPage[nextOverFlowPageNumFie
ldOffset] }, 0);

getDataFromOverflowPages(nextOverf
lowPage, ref ptr, ref schema,
colNum, ref recordHeaderSize);
ptr = ptr + keyValueFieldLength;

long
min_embeded_fraction=headerBytesOf
SQLiteFile[22];
long max_local = pageSize - 35;
long min_local = (pageSize - 12) *
min_embeded_fraction / 255 - 23;
long local_size = min_local +
(recordSizeValue - min_local) %
(pageSize - 4);
if (local_size > max_local)
local_size = min_local;
long
nextOverFlowPageNumFieldOffset =
ptr + local_size;
Dictionary<int, ArrayList> schema
= new Dictionary<int,
ArrayList>();


if (recordSizeValue <=
max_local)// small records
{
long recordHeaderSize = 0;
nextOverFlowPageNumFieldOffset =
pageSize;
try
{
    Array.Copy(currentPage, ptr,
buffer, 0, 9);
}
catch (ArgumentException ex)
{
    Array.Copy(currentPage, ptr,
buffer, 0, currentPage.Length -
ptr);
}

int index =
Utils.varInt2Int(buffer, ref
recordHeaderSize);
ptr = ptr + index;
recordHeaderSize =
recordHeaderSize - index;

for (int i = 0;
recordHeaderSize!=0; i++)
{
    if (ptr==260)
        Debug.Write("");
```

```
        ptr = pageSize;

        isOverflowPage = true;
    }
}
if (isOverflowPage)
{
    do
    {

        if (pageSize - ptr >=
buffer.Length - bufPtr)
        {

Array.Copy(currentPage, ptr,
buffer, bufPtr, buffer.Length -
bufPtr);
            bufPtr =
buffer.Length;
            ptr = ptr +
buffer.Length - bufPtr;
            break;
        }
        else
        {

Array.Copy(currentPage, ptr,
buffer, bufPtr, pageSize - ptr);
            bufPtr = bufPtr +
pageSize - ptr;
            ptr = pageSize;
        }
        currentPage =
Utils.ReadingFromFile(dbFilePath,
(nextOverflowPage - 1) * pageSize,
pageSize);
        nextOverflowPage =
BitConverter.ToInt32(new byte[] {
currentPage[3], currentPage[2],
currentPage[1], currentPage[0] },
0);
        ptr = 4;
    } while (bufPtr !=
buffer.Length);
}

string value = "";
byte[] buf;
switch (type)
{
    case "STRING":
}
}
if (schema.Count == 1)
{
Debug.WriteLine("");
}
Dictionary<int,string>
currentRecord=new
Dictionary<int,string>();
for (int i = 0;
schema.ContainsKey(i); i++)
{
ArrayList item = schema[i];
string type = (string)item[0];
long colLength =
Convert.ToInt64(item[1]);
buffer = new byte[colLength];
int bufPtr = 0;
nextOverflowPage =
BitConverter.ToInt32(new byte[] {
currentPage[3], currentPage[2],
currentPage[1], currentPage[0] },
0);

if (!isOverflowPage)
{
    if (colLength + ptr <=
nextOverFlowPageNumFieldOffset)
    {
        Array.Copy(currentPage,
ptr, buffer, 0, colLength);
        ptr = ptr +
(int)colLength;
        bufPtr = buffer.Length;
    }
    else
    {
        Array.Copy(currentPage,
ptr, buffer, 0,
nextOverFlowPageNumFieldOffset -
ptr);
        bufPtr = pageSize - ptr;
        nextOverflowPage =
BitConverter.ToInt32(new byte[] {
currentPage[nextOverFlowPageNumFie
ldOffset + 3],
currentPage[nextOverFlowPageNumFie
ldOffset + 2],
currentPage[nextOverFlowPageNumFie
ldOffset + 1],
currentPage[nextOverFlowPageNumFie
ldOffset] }, 0);
```

```
        break;
    default:
        value =
BitConverter.ToString(buffer);
        break;
}
currentRecord.Add(i,value);
}
records.Add(currentRecord);

}

private void
getDataFromOverflowPages(int
nextOverflowPage, ref int ptr, ref
Dictionary<int, ArrayList> schema,
int colNum, ref long headerSize) {
if (headerSize == 0)
{
return;
}
else
{
currentPage =
Utils.ReadingFromFile(dbFilePath,
(nextOverflowPage - 1) * pageSize,
pageSize);
nextOverflowPage =
BitConverter.ToInt32(new byte[] {
currentPage[3], currentPage[2],
currentPage[1], currentPage[0] },
0);
byte[] buffer = new byte[9];
ArrayList item =
extractCurrentColLength(ref ptr,
buffer, pageSize, ref headerSize);
schema.Add(colNum, item);
getDataFromOverflowPages(nextOverf
lowPage,ref ptr, ref
schema,colNum++,ref headerSize);
}

}
/// <summary>
///
/// </summary>
/// <param name="ptr"></param>
/// <param name="buffer"></param>
/// <param
name="nextOverflowPageOffset"></pa
ram>
        value =
Encoding.UTF8.GetString(buffer);
        break;
    case "FLOAT":
        value =
Convert.ToString(BitConverter.ToDo
uble(buffer, 0));
        break;
    case"INTEGER_0":
        value = "-";
        break;
    case "INTEGER_1":
        buf=new byte[2];
        Array.Copy(buffer, 0, buf,
0, buffer.Length);
        value =
Convert.ToString(BitConverter.ToIn
t16(buf, 0));
        break;
    case "INTEGER_2":
        value =
Convert.ToString(BitConverter.ToIn
t16(buffer, 0));
        break;
    case "INTEGER_3":
        buf = new byte[4];
        Array.Copy(buffer, 0, buf,
0, buffer.Length);
        value =
Convert.ToString(BitConverter.ToIn
t32(buf, 0));
        break;
    case "INTEGER_4":
        value =
Convert.ToString(BitConverter.ToIn
t32(buffer, 0));
        break;
    case "INTEGER_6":
        buf = new byte[8];
        Array.Copy(buffer, 0, buf,
0, buffer.Length);
        value =
Convert.ToString(BitConverter.ToIn
t64(buf, 0));
        break;
    case "INTEGER_8":
        value =
Convert.ToString(BitConverter.ToIn
t64(buffer, 0));
        break;
    case "NULL":
        value = null;
```

```
/// <param
name="headerSize"></param>
/// <returns>Array list of 2
string first one is type and the
second one is length.</returns>
private ArrayList
extractCurrentColLength(ref int
ptr, byte[] buffer, long
nextOverflowPageOffset,ref long
headerSize)
{
if (ptr == 260)
Debug.Write("");
long typeNValue = 0;

if (nextOverflowPageOffset == 0)
{
if (ptr + 9 < pageSize)
    Array.Copy(currentPage, ptr,
buffer, 0, 9);
else
    Array.Copy(currentPage, ptr,
buffer, 0, pageSize - ptr);
}
else
{
if (ptr + 9 <
nextOverflowPageOffset)
    Array.Copy(currentPage, ptr,
buffer, 0, 9);
else
{
    Array.Copy(currentPage, ptr,
buffer, 0, nextOverflowPageOffset
- ptr);
    byte
[]page=Utils.ReadingFromFile(dbFil
ePath,(nextOverflowPageOffset-
1)*pageSize,pageSize);
    Array.Copy(page, 5, buffer,
nextOverflowPageOffset - ptr, 9 -
(nextOverflowPageOffset - ptr));
}
}

int index =
Utils.varInt2Int(buffer, ref
typeNValue);
ptr = ptr + index;
headerSize = headerSize - index;
long colLength = 0;
string type = "";
```

```
ArrayList item=new ArrayList();
if (typeNValue > 11 && typeNValue
% 2 == 0)
{
colLength = (typeNValue - 12) / 2;
type = "BLOB";
item.Add(type);
item.Add(colLength);
}
else if (typeNValue > 11 &&
typeNValue % 2 == 1)
{
colLength = (typeNValue - 13) / 2;
type = "STRING";
item.Add(type);
item.Add(colLength);
}
else
{
try
{
    item =
sqliteTypes[Convert.ToInt16(typeNV
alue)];
}
catch (OverflowException ex)
{
    Debug.WriteLine("");
}
}
return item;
}

internal void WALFileParser()
{
throw new
NotImplementedException();
}
}
```

در ادامه کلاس بازیابی بـه کمـک فایـل ژورنـال آمده است.

```
internal class JournalFileParser
{
private string journalFilePath;
private long pageSize;
private long sectorSize;
private const int
journalHeaderLength = 28;
```

```
differences of old db with current
one.
/// </summary>
/// <returns>records whitch is
omited or updated from old db
classified by table
name.</returns>
private Dictionary<string,
ArrayList> findDeletedRecords()
{
List<long> keys =
backupPages.Keys.ToList<long>();
for (int i = 0; i < maxListLength;
i++)
{

System.IO.File.Copy(dbFilePath,
rollbackedFile+"_"+i, true);
    Stream outStream =
File.Open(rollbackedFile+"_"+i,
FileMode.Open);
    for (int j = 0; j <
keys.Count; j++)
    {
        long pageNum = keys[j];
        ArrayList list =
backupPages[pageNum];
        if (list.Count != 0)
        {
            byte[] currentPage =
new byte[pageSize];
            currentPage =
Utils.ReadingFromFile(journalFileP
ath, (long)list[0],
(int)pageSize);

outStream.Seek((pageNum - 1) *
pageSize, SeekOrigin.Begin);

outStream.Write(currentPage, 0,
currentPage.Length);
            list.RemoveAt(0);
            backupPages[pageNum] =
list;
        }
    }
    outStream.Flush();
    outStream.Close();
}

Dictionary<string,
Dictionary<string, ArrayList>>
private byte[] journalMagic = new
byte[] { 0xd9, 0xd5, 0x05, 0xf9,
0x20, 0xa1, 0x63, 0xd7 };
private const int checksumLength =
4;
private Dictionary<long,
ArrayList> backupPages = new
Dictionary<long, ArrayList>();
private string dbFilePath;
private long recordsCount;
private long fileSize;
private int maxListLength = 0;
private string path;
private string rollbackedFile;
private Dictionary<string,
ArrayList> records;

internal JournalFileParser(string
journalFilePath, string
dbFilePath, string workSpacePath)
{

this.journalFilePath =
journalFilePath;
this.dbFilePath = dbFilePath;
this.path = workSpacePath;
rollbackedFile = workSpacePath +
@"\rollBackedFile";
if
(!Directory.Exists(workSpacePath))
{

Directory.CreateDirectory(workSpac
ePath);
}

init();

fillBackupPages();

records = findDeletedRecords();

}

internal Dictionary<string,
ArrayList> getDeletedRecords()
{
return records;
}
/// <summary>
/// Build old db from journal
pages and then find the
```

```
    for (var i = 0; i <
dr.FieldCount; i++)
    {

colNames.Add(dr.GetName(i));
    }
    records.Add(colNames);


    do
    {
        ArrayList mQuery =
queries[tableName][filePathes[inde
x]];
        foreach (string query in
mQuery)
        {
            SQLiteCommand com =
new SQLiteCommand(query,
connection);
            SQLiteDataReader
reader = com.ExecuteReader();

            while (reader.Read())
            {
                ArrayList item =
new ArrayList();
                foreach (string
col in colNames)

item.Add(reader[col]);
                records.Add(item);
            }
        }

Utils.closeSqlitConnection(connect
ion);

        index++;
        if (index <
filePathes.Length)
            connection =
Utils.buildDBConnection(filePathes
[index]);
    } while (index <
filePathes.Length);
    if (records.Count > 1)
        result.Add(tableName,
records);
}
return result;
```

```
result = new Dictionary<string,
Dictionary<string, ArrayList>>();
for(int i=0;i<maxListLength;i++){

Utils.getDataBaseDifferences(rollb
ackedFile + "_" + i,
dbFilePath,ref result);
}
return getRecords(result);

}
/// <summary>
/// retrieved records from old dbs
with given queries.
/// </summary>
/// <param name="queries">contaion
queries catagorized by table name
at first and then by db
path.</param>
/// <returns>records that
classified by table
names.</returns>
private static Dictionary<string,
ArrayList>
getRecords(Dictionary<string,
Dictionary<string, ArrayList>>
queries)
{
Dictionary<string, ArrayList>
result = new Dictionary<string,
ArrayList>();

foreach (string tableName in
queries.Keys)
{
    ArrayList records = new
ArrayList();

    string[] filePathes =
queries[tableName].Keys.ToArray();
    int index = 0;
    SQLiteConnection connection =
Utils.buildDBConnection(filePathes
[index]);

    var cmd = new
SQLiteCommand("select * from " +
tableName, connection);
    var dr = cmd.ExecuteReader();
    ArrayList colNames = new
ArrayList();
```

```
    }
    offset = offset + pageSize +
checksumLength;
}

}

private void init()
{
fileSize =
Utils.fileSize(journalFilePath);
byte[] header =
Utils.ReadingFromFile(journalFileP
ath, 0, journalHeaderLength);

sectorSize =
BitConverter.ToInt32(new byte[] {
header[23], header[22],
header[21], header[20] }, 0);
pageSize =
BitConverter.ToInt32(new byte[] {
header[27], header[26],
header[25], header[24] }, 0);

if (sectorSize == 0)
    sectorSize = 512;
if (pageSize == 0)
{
    byte[] result =
Utils.ReadingFromFile(dbFilePath,
16/*pageSizeOffsetValue*/,
2/*pageSizeLengthValue*/);
    Array.Reverse(result);
    pageSize =
BitConverter.ToInt16(result, 0);
}

}
}
```

در ادامه کد پیاده‌سازی شده برای افزونه اندروید آمده است.

```
using System;
using System.Collections.Generic;
using System.Linq;
using System.Text;
using DevicePluginInterface;
using System.Windows.Forms;
using RegawMOD.Android;
using System.Diagnostics;
}

/// <summary>
/// find offset of leaf pages in
journal file.
/// </summary>
private void fillBackupPages()
{
long offset = sectorSize;


while
(fileSize/sectorSize!=offset/secto
rSize)
{
    byte[]
pageNumArray=Utils.ReadingFromFile
(journalFilePath, offset, 5);
    int pageType=pageNumArray[4];
    long currentPageNumber =
BitConverter.ToInt32(new byte[] {
pageNumArray[3], pageNumArray[2],
pageNumArray[1], pageNumArray[0]
}, 0);
    offset=offset+4;

    if (pageType == 13)
    {
        if
(backupPages.ContainsKey(currentPa
geNumber))
        {
            ArrayList list =
backupPages[currentPageNumber];
            list.Add(offset);

backupPages[currentPageNumber] =
list;
            if (list.Count >
maxListLength)
                maxListLength =
list.Count;
        }
        else
        {
            ArrayList list = new
ArrayList();
            list.Add(offset);

backupPages.Add(currentPageNumber,
list);
        }
```

```
public void
ConnectingDevice(object sender,
EventArgs e)
{
device =
android.GetConnectedDevice();
}

public void
copyAppDataBaseFromDevice(string
key, string path, string
destination)
{
if (!Directory.Exists(destination
+ key))
{
Directory.CreateDirectory(destinat
ion + key);
}

AdbCommand adbCmd =
Adb.FormAdbShellCommand(device,
true, "chmod", new object[] { 777,
path });
Debug.WriteLine(Adb.ExecuteAdbComm
and(adbCmd));
adbCmd =
Adb.FormAdbShellCommand(device,
true, "chmod", new object[] { 777,
path + "-journal" });
Debug.WriteLine(Adb.ExecuteAdbComm
and(adbCmd));

device.PullFile(path, destination
+ key);
device.PullFile(path + "-journal",
destination + key);

}

public bool isDeviceRoot()
{
return device.HasRoot;
}


public bool isDeviceConnected()
{
return
android.HasConnectedDevices;
}
```

```
using System.Threading;
using System.IO;

namespace AndroidPlugin
{
public class Plugin :
DeviceRecoveryPluginInterface
{
private AndroidController android;
private Device device;
private Thread thread;

public Plugin()
{
android =
AndroidController.Instance;
DeviceConnect += ConnectingDevice;
if (!android.HasConnectedDevices)
{
thread = new
Thread(waitForDevice);
thread.Start();
}
else
{
device =
android.GetConnectedDevice();
}
}
private void waitForDevice()
{
android.WaitForDevice();
OnDeviceConnected(EventArgs.Empty)
;

thread.Abort();
}
protected virtual void
OnDeviceConnected(EventArgs e)
{
EventHandler handler =
DeviceConnect;
if (handler != null)
{
handler(this, e);
}
}

public EventHandler DeviceConnect;
```

```
public bool rootDevice()
{
throw new
NotImplementedException();
}

public bool unRootDevice()
{
throw new
NotImplementedException();
}

public void refreshDeviceList()
{
android.UpdateDeviceList();
}


public bool installApp(string
path)
{
return device.reInstallApk(path);
}
}
}
```

Amirkabir University of Technology
(Tehran Polytechnic)

Computer and Information Technology Engineering Department

B.Sc. Thesis

Title

Design and Implementation of Extensible Software in order to Retrieve Deleted Information from Smart Phones

By
Ehsan Edalat

Supervisor
Dr. Babak Sadeghian

September 2015