به نام خدا

دانشگاه آزاد اسلامی واحد علوم و تحقیقات ایلام

گروه مهندسی کامپیوتر

موضوع:

تامین امنیت تاریخچه:

امنیت و پاسخگویی در سیستم های بانک اطلاعاتی

(Securing history: Privacy and accountability in database systems)

نویسندگان:

Gerome Miklau, Brian Levine, Patrick Stahlberg

University of Massachusetts

مترجم:

امیر حسین رحیمی

شماره دانشجویی: ۹۱۰۸۵۰۲۶۳

زمستان ۱۳۹۱

چکیده:

پایگاه های داده ای که یک مجموعه از سوابق عملیات و فعالیت های خود را نگهداری می کنند دارای قابلیت پاسخگویی خوبی هستند: وقایع و رخداد های گذشته جهت کشف خطا و حفظ کیفیت داده ها می توانند تجزیه و تحلیل شوند. اما نگهداری سوابق می تواند تهدیدی برای امنیت اطلاعات و حفظ حریم خصوصی باشد. طراحان سیستم باید به دقت حفظ حریم خصوصی و پاسخگویی را از طریق کنترل اینکه چه زمانی و چگونه سیستم داده را ذخیره و چه کسی قادر به بازیابی و تجزیه و تحلیل آنها باشد متعادل کند. این مقاله چالش های تکنیکی فراروی امنیت و حفظ حریم خصوصی سیستم های بانک اطلاعاتی را زمانی که سوابق را به طور ایمن نگهداری می کند توصیف می کند. این چالش ها شامل: یک، تشخیص نگهداری ناخواسته داده در زمان حیات سیستم بانک اطلاعاتی که می تواند امنیت را تهدید کند؛ دوم، طراحی دوباره اجزاء سیستم برای اجتناب از ذخیره ناخواسته داده؛ سوم، توسعه خصوصیات جدید سیستم جهت پشتیبانی از پاسخگویی زمانی که مورد نیاز است.

۱. معرفی

به دلیل اینکه خطاها و نا هنجاری ها به طور کامل اجتناب پذیر نیست بسیاری از برنامه های کاربردی که داده های حساس را مدیریت می کنند، یک سابقه از فعالیت ها و داده ها را نگهداری می کنند. این کار پاسخگویی را فراهم می کند چون وقایع گذشته می توانند جهت کشف خطا، حفظ کیفیت داده ها و بازبینی خط مشی های امنیتی تجزیه و تحلیل شود. برای مثال پاسخگویی در یک سیستم مدیریت اطلاعات دارویی بسیار سخت است. اگر اطلاعات غلط از یک رکورد دارویی استخراج شود، این مهم است وقتی که خطایی پیش بیاید چه کسی در برابر آن مسئول است و چگونه این داده های غلط ممکن است استفاده شود. حافظه های ارزان ذخیره تمام پایگاه های داده گذشته را فراهم می کنند و بسیاری معتقدند ارتقا یا نگارش نسخه های جدید پایگاه های داده باید یک قانون شود نه استثناء [48,49,2,29].

در جایی که بحث ذخیره سوابق داده ها یا عملیات گذشته مطرح است هنوز زمینه هایی برای تهدید امنیت و اعتماد پایگاه داده ها وجود دارد، برای مثال در سازمان ها و شرکت های حساس که اطلاعات بسیار حساس را نگهداری می کنند (با ریسک بسیار بالا به دلایل امنیتی یا به اجبار) و ممکن است قوانین حفاظت اطلاعات نقض شود مجبور به حذف به موقع داده ها می شود[32, 25, 23]. به طور مشابه بدون کنترل دقیق در پاک کردن داده ها آثار و بقایای به جا مانده داده های گذشته می تواند منجر به بروز مشکل جدی شود. برای مثال محققان داده های بسیار با ارزشی را پس از باز بینی نمونه ای از هارد درایو های مجاز از داده های حساس جمع آوری کرده اند[27]. اسناد دیجیتالی ساخته شدند تا داده های حساس را نگهداری کنند که باید به طور ایمن حذف شوند[14 ,24]. در موارد قانونی پست الکترونیک علیه کارمندان Enron استفاده و عمومیت یافت[30, 33, 47].

سیستم هایی که سوابق را نگهداری نمی کنند می توانند امنیت خود را حفظ کنند . آنها مقادیر واقعی را در تنظیمات مشخصی دارند. Rosen [44] شواهد و مدارکی را در مورد یک داروخانه قدیمی در واشینگتون گزارش می دهد. این داروخانه هیچ داده کامپیوتری نداشت و می توانست به موفقیت عظیمی در تولید داروی ضد افسردگی Viagra و سایر داروهای حساس و مهم که جنبه سیاسی داشتند دست یابد. علاوه بر آن قانون قضایی مینه سوتا معتقد است که استفاده از داده های حذف شده و بازیابی شده غیر قانونی است[45].

واقعیت این است که حفظ حریم خصوصی و پاسخگویی هر دو مهم اند، اما آنها اغلب با هم در تضادند و طراحان سیستم باید به دقت بین این دو تعادل برقرار کنند. موضوع اصلی این است که چه زمانی و چگونه داده توسط سیستم بازیابی شده و چه کسی قادر به بازیابی و آنالیز آن باشد. این مقاله در مورد بهبود کارایی پایگاه های داده بحث می کند که به کاربران اجازه می دهد با دقت سوابق را مدیریت و بین

پاسخگویی و امنیت تعادل برقرار کند. تنظیماتی که به آن نیاز است، پایگاه داده باید به عنوان یک سیستم با حافظه کم تنظیم شود که امنیت پایگاه را بوسیله رد کردن تلاش های غیر مجاز برای دستیابی به پایگاه داده جهت ردیابی فعالیت ها یا بازیابی داده های حذف شده حفظ کند. تنظیم دیگری که به آن نیاز است این است که پایگاه داده باید پاسخگویی را بوسیله نگهداری سوابق، اجازه دادن به آنالیز آن و کنترل دستیابی پشتیبانی کند. بدبختانه همانطور در زیر توصیف می کنیم در پایگاه های داده موجود تعادل بین پاسخگویی و امنیت برقرار نیست.

تهدیدات در برابر حریم خصوصی

پایگاه های داده موجود حریم خصوصی و محرمانگی کاربران را به دلیل نگهداری سهوی داده ها و عملیات تهدید می کنند. یک پایگاه داده فعال از داده های حساس کپی های متعددی در جداول ذخیره سازی، فهرست، گزارش یا log ، شمای خارجی و روابط موقت تهیه می کند. مدیر جدول ذخیره سازی رکوردهای پایگاه داده را در فضای تخصیص یافته ذخیره و همچنین یک کپی از رکورد های داده را در فضای سیستم فایل معمولی ذخیره می کند. زمانی که داده ها حذف می شوند کاملا از بین نمی روند و هنوز بر سطح دیسک باقی می مانند، صاحبان داده در حال حاضر روی این گونه عملیات یا کنترل کم یا هیچ گونه کنترلی ندارند و نمی توانند به طور قطع بگویند که داده های حساس حذف شده اند، آیا آنها پس از حذف کاملا از بین رفته اند یا چه مدت بر سطح دیسک باقی می مانند.

یک زمینه نو ظهور در مباحث قانونی کامپیوتر به بحث آنالیز سیستم ها می پردازد به عبارت دیگر فرضیات را در مورد فعالیت های گذشته ارزیابی می کند. یک آنالیز قانونی از سیستم پایگاه داده می تواند سوابق جزئی از آثار ناخواسته ذخیره شده توسط سیستم را تولید کند. به طور معمول بازرس قانونی یک مجری حرفه ای قانونی است که یک کامپیوتر در تصرف اوست و دسترسی بدون محدودیت به دیسک را دارد. اما یک آنالیز قانونی می تواند چه از طریق سرقت یا از دست رفتن اطلاعات توسط طیف وسیعی از دشمنان نیز انجام شود مانند: هکرها، کاربران خودی ممتاز یا هر کسی که دسترسی فیزیکی به سخت افزار دارد. مادامی که رمزگذاری داده می تواند به حفاظت در برابر دستیابی های غیر مجاز کمک کند، ممکن است نتواند جلوی کاربران خودی را بگیرد که این خود اغلب تهدیدی برای امنیت و حفظ حریم خصوصی است [54]. کاربران در مورد حریم خصوصی خود نگرانند که باید تهدیدهایی در نتیجه تحلیل قانونی ممکن است پیش بیاید باید در نظر گرفته شود.

پاسخگویی نامناسب

نگهداری نا خواسته سوابق اطلاعات موضوعی فراگیر در مورد پایگاه های داده است، اما این کار به منظور افزایش پاسخگویی در برابر کاهش امنیت انجام می گیرد. اولا هیچ راهی وجود ندارد که تضمین کند اطلاعات مورد نیاز در دسترس خواهد بود (بخش ۲ را ببینید). دوما یک عدم توازن بین امنیت و پاسخگویی وجود دارد. یک مقدار داده ای حساس نگهداری و بازیابی شده از یک محتوای نامناسب می تواند امنیت را نقض کند. به عنوان مثال یک کد امنیتی عمومی یا یک تشخیص طبی. اما افراد مقادیری را ذخیره و نگهداری می کنند که برای پاسخگویی مناسب نیستند و اغلب نیاز به تکمیل اسناد و مدارک کامل گذشته را دارند.

سیستم های بانک اطلاعاتی موجود شامل مجموعه ای از اجزا هستند که جهت نگه داری آگاهانه سوابق طراحی شده اند. لوگ یا گزارش تراکنش ها، تمام تغییرات دیتابیس را ذخیره می کند، پشتیبان ها ذخیره و نگه داری می کند و بسیاری از سیستم ها شامل یک لوگ یا گزارش عملیات است که موارد مشکوک را که باعث بروز مشکل شده اند، عملیات کنترل دستیابی و نظارت بر عملیات سیستمی در میان سایر کارها می کند. اضافه به آن محققان طیف وسیعی از مکانیزم های نگارش و آرشیو کردن برای دیتابیس را پیشنهاد داده اند که می تواند سوابق کامل را نگهداری و بتواند قابلیت های جستجوی دیتابیس را از حالت های گذشته تحقیق کند[40, 51, 50, 11, 34, 35].

اما این تکنولوژی ها محدودیت هایی نیز دارند، لوگ تراکنش ها و پشتیان ها نمی تواند به طور خوب یک ابزار کارآمد برای پرس و جو از میان حالت های گذشته دیتابیس

باشد. آرشیو کردن و دیتابیس های موقتی در عمل به طور وسیع مورد استفاده نیستند و اگر ارتقا مورد نیاز باشد این کار معمولا در میان افزار انجام می گیرد، علاوه بر این در تمام این سیستم ها مکانیزم های کنترل دستیابی به لوگ و رکورد های سوابق،که جهت افزایش پاسخگویی است بسیار محدودند. داده های سوابق و لوگ کاملا خارج از مکانیزم های سنتی کنترل دستیابی به پایگاه داده است. روی هم رفته، سیستم های موجود فاقد کارآمدی و قابل کانفیگ برای نگهداری رضایت بخش سوابق، آنالیز اطلاعات برای فراهم آوردن پاسخگویی مناسب و کنترل دستیابی و نگهداری آن اطلاعات برای اهداف امنیتی نیستند.

مشارکت

در ادامه مقاله تلاش های فعلی ما در مهندسی پایگاه های داده جهت مدیریت تاریخچه به صورت ایمن توصیف می شود.

ما با توصیف تهدید ها علیه امنیت شروع می کنیم که این نتیجه تحلیل قانونی سیستم های پایگاه داده است کار مقدماتی ما [37] تحقیق در مورد چهار پایگاه داده معروف و یک پایگاه داده کتابخانه ای جاسازی شده است، این چهار تا عبارتند از: (SQLite, IBM, DB2, MySQL). نشان می دهیم در این موارد ذخیره سازی داده ها در جداول واقعا تعجب برانگیز است. تاپل های حذف شده مدتها پس از آنکه حذف شده اند اغلب می توانند بازیابی شوند. اضافه بر این مقادیر داده ای دوره حیات پیچیده ای دارند. نه تنها فایل های اختصاص داده شده به پایگاه داده ، بلکه همچنین در فایل سیستم که خارج از کنترل پایگاه داده است ذخیره شده اند. ما یافته های اصلی مان را از کار مقدماتی به طور کوتاه در بخش۲ این بحث را خلاصه کرده ایم [37].

این آنالیز قانونی نشان می دهد که رابط استاندارد به پایگاه داده (برای مثال SQL) به طور قابل اعتماد محتوای واقعی پایگاه داده را نشان نمی دهد. چنانچه قطع ارتباط بین رابط و سیستم نگرانی جدیی برای صاحبان داده است چون که سیستم یک شمای غلط از نگه داری داده را نشان می دهد . ما این موضوع را با طراحی یک سیستم جدید به نام شفافیت در بخش ۳ بررسی می کنیم. به طور خلاصه تمام داده هایی که در سیستم نگه داری شده است باید قابل دست یابی از طریق یک رابط مشروع (قانونی) باشد و نباید بازیابی داده های پنهان از طریق بازرسی وضعیت سیستم ممکن باشد. در مواردی که داده ها فراتر از طول عمر فعال خود نگهداری می شوند، تداوم آن باید در سیستم آشکار و به دقت قابل تنظیم باشد. ما به توصیف چالش های فنی پیرامون اصلاح اجزای سیستم برای شفافیت می پردازیم (جدول ذخیره سازی، ایندکس هاف و لوگ گزارشات). این چالش ها شامل پیاده سازی ایمن، حذف و ساخت ساختمان های داده مستقل از تاریخچه است.

سرانجام در بخش ۴ ما به توصیف چالش های کلیدی در بوجود آوردن ویژگی های پاسخگویی در پایگاه های داده می پردازیم. اهداف اصلی جمع آوری داده های مناسب از طریق مکانیسم های پایداری برای اجازه دادن آنالیز داده از از طریق پردازش پرس و جوی کارآمد، و به طور مهم برای حفاظت از داده ها از طریق کنترل دوره های نگهداری و گروه هایی که می توانند به آن دستیابی داشته باشند. ما همچنین برای بهبود حفاظت از تاریخچه یک عملیات ویرایش تاریخچه را توصیف خواهیم کردکه برای پاک کردن تمام داده های حساس گذشته از پایگاه داده استفاده می شود. ما کار مربوط به هر یک از این موضوعات را در داخل بخش مربوطه توصیف خواهیم کرد.

۲. تحلیل قانونی پایگاه های داده

علوم قضایی کامپیوتر یک علم نو ظهور [18] است که بازیابی داده از سیستم های فایل [17, 28, 26, 27]، نگهداری ناخواسته داده توسط برنامه هایی مانند مرورگر های وب و فایل های اسناد را آموزش می دهد [26]. ابزار های قانونی مانند: Sleuth Toolkit [16] و EnCASE Forensic [22] رایج ترین ابزارها برای بازرسان جهت بازیابی داده ها از سیستم های کامپیوتری است. این ابزارها بعضی اوقات قادرند تا فایل های رایج را ترجمه و تفسیر کنند اما برای نیاز ما پشتیبانی لازم را برای تحلیل فایل های پایگاه داده را فراهم نمی کند.

تحلیل گران قانونی معمولا به طور نا محدود به فضای ذخیره سازی دیسک دستیابی دارند. ما مدل های تهدید امنیتی را با نمونه هایی از انواع دستیابی بررسی می کنیم، به عنوان مثال استفاده از قابلیت های مدیران سیستم، یک هکر که اجازه دستیابی ممتاز به سیستم را بدست آورده است، و یا یک مهاجم که امنیت فیزیکی را نقض کرده و به سیستم دستیابی پیدا کرده است. ما همچنین باید توجه داشته باشیم که پایگاه های داده به صورت فزاینده ای در داخل طیف وسیعی از برنامه هایی مقاوم برای امنیت جاسازی شده اند به عنوان مثال کتابخانه های پایگاه داده های جاسازی شده مانند: BerkeleyDB [5] و SQLite [46] که تحت مکانیزم های ذخیره سازی زیرین برای مشتریان ایمیل، مرورگرهای وب، پیاده سازی LDAP و Google Desktop مورد استفاده واقع می شود.به عنوان مثال برنامه پست الکترونیکی Apple از یک پایگاه داده SQLite برای پشتیبانی از جستجو در موضوعات، گیرنده و فرستنده استفاده می کند. اخیرا Mozilla، SQLite را به عنوان یک مدل ذخیره سازی یکپارچه برای تمام اپلیکیشن های خود پذیرفته است. در FirFox 2.0 سایت های راه دور می توانند داده های ایمن در یک پایگاه داده SQLite را به عنوان جایگزین پیچیده ای از کوکی ها ذخیره کنند. تحلیل قانونی بعضی حافظه های جاسازی شده واقعا جالب است چون آنها هر روز زیر فشار برنامه های کاربردی کاربران خانگی هستند و همچنین حفاظت از پایگاه های داده جاسازی شده در برابر تهدیدات سخت تر است.

تحلیل قانونی می تواند در مورد بسیاری از اجزای سیستم های پایگاه داده استفاده شود، و این نه تنها داده های جاری فعال پایگاه داده بلکه داده های حذف شده و سوابق تاریخی در مورد عملیات انجام شده سیستم را نیز آشکار می کند. حافظه رکورد در پایگاه داده شامل داده هایی است که به طور منطقی حذف شده اند اما کاملا از بین نرفته اند. همچنین ایندکس ها شامل مقادیر حذف شده ای هستند اضافه بر این ممکن است نشان داده شود از طریق ساختار آنها اثرات عملیات گذشته منجر به بروز چنین حالت هایی می شود. طبیعتا لوگ تراکنش ها شامل اطلاعات قانونی زیادی از تصویر ماقبل و مابعد هر بروز رسانی پایگاه داده است. منابع دیگر اطلاعات قانونی شامل روابط موقت (اغلب برای مرتب سازی در حجم عظیم عملیات نوشته شده است)، کاتالوگ پایگاه داده، شناسه تاپل های مخفی که ممکن است منظور از ایجاد تاپل ها را در پایگاه داده نشان دهد. در اینجا هدف فهمیدن مقدار نگهداری داده در حافظه و همچنین طول دوره حیات مورد انتظار داده است.

ما باید توجه داشته باشیم که حافظه رمزگذاری شده اغلب برای مبارزه با این تهدیدات کافی نیست و به عنوان یک موضوع عملی حافظه های رمزگذاری شده به طور وسیع مورد استفاده نیستند. در پایگاه های داده رمزگذاری اغلب منجر به هزینه های عملکردی غیر قابل قبول می شود، اضافه بر این تحلیل گران قانونی یا مهاجمان ممکن است کلید مخفی را بازیابی کنند، چون این کلید توسط کرابران زیادی به اشتراک گذاشته، به راحتی احضار و روی دیسک ذخیره و بازیابی می شود.

۲,۱ تحلیل قانونی جدول ذخیره سازی

برای شروع، ما ذخیره سازی نا خواسته داده در پایگاه داده واقعی شامل PostgreSQL, MySQL, IBM DB2 و پایگاه داده رایج جاسازی شده SQLite را مطالعه می کنیم.

مشاهده اول ما این است که در تمام سیستم هایی که مطالعه می کنیم حذف رکورد ایمن نیست. حذف رکورد با یک حذف جزئی که داده روی آن مجددا نوشته نشده کاملا قابل بازیابی است. این اطلاعات قابل بازیابی منقضی شده اند (در نتایج جستجو ظاهر نمی شوند) که ما آن را database slack و به اختصار DB-Slack می خوانیم.

طول عمر DB-Slack در موراد حذف رکورد مشکل است که پیشگویی شود. پس از حذف فضایی که توسط رکورد اشغال شده بود آزاد می شود و ممکن است توسط یک رکوردی که درج شود دوباره گرفته شود. اسفاده مجدد از فضای آزاد شده توسط رکورد جدید وابسته به این است که رکورد جدید جهت درج در فضای آزاد فیت و مناسب باشد (از ویژگی های رکورد ممکن است طول متغیر باشد) و اینکه آیا نظم و ترتیب در جدول بوسیله محدودیت های خوشه بندی رعایت شده است. اضافه بر این چون صفحات در

حافظه رکورد با گذر زمان تکه تکه[1] می شوند در اینجا یک دستور تنظیم مجدد جدول وجود دارد (مراجعه جهت خلاء کردن[2]) که به صورت دوره ای توسط مدیران سیستم اجرا می شود. وقتی vacuum اجرا شود رکورد ها در داخل و سراسر صفحات کپی می شوند و فضای اشغال شده توسط فایل برای جدول ممکن است کاهش یابد. بنابراین فضا به سیستم فایل بازگردانده می شود.

برنامه vacuum می تواند به کاهش قابلیت بازیابی داده ها به وسیله رو نویسی مجدد DB-slack کمک کند اما ما اثر نمایان دیگری را vacuum کشف کرده ایم که آن افزایش قابلیت بازیابی داده است. این یک پیامد و دستاورد دوم از کشفیات ما این است که عملیات vacuum نا امن است. تمام سیستم هایی که ما امتحان کردیم فضا را به سیستم فایل بدون رونویسی مجدد داده بازگردانده اند. این بدان معناست که کپی هایی از رکورد های پایگاه داده به فضای تخصیص نیافته و آزاد سیستم فایل انتقال داده شده و قابل بازیابی باقی ماندند. داده هایی که در سیستم فایل پس از پاک کردن باقی می مانند file system slack و یا به اختصار FS-Slack خوانده می شوند.

تفاوت بین DB-Slack و FS-Slack در فایل اختصاص داده شده به پایگاه داده وجود دارد. پس با روش پاکسازی کامل توسط سیستم فایل از بین نمی روند و این توسط یک سیستم فایل مجهز به حذف ایمن حل نمی شود. ما در بخش ۳ در مورد حذف داده های به جا مانده بحث خواهیم کرد.

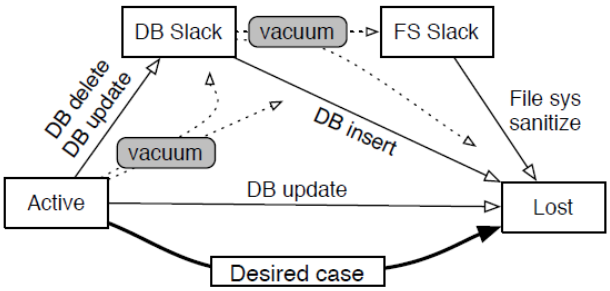
ردیابی عمر تاپل در جدول ذخیره سازی

درک طول عمر مقادیر داده ای در پایگاه های داده برای تضمین امنیت پایگاه مهم و حیاتی است، نمودار حالات در شکل ۱ به صورت تصویری جریان پیچیده داده در یک بانک اطلاعاتی را در زمان حیات آن نشان می دهد. داده ها با درج در پایگاه داده در حالت فعال حیات خود را آغاز می کنند. در حالت ایده آل داده های فعال باید پس از حذف از از بین بروند، با بررسی حالت گذار در پایین حالات این حالت وقتی اتفاق می افتد که تاپل ها چند بار آپدیت شده و داده ها مجددا رونویسی شوند، اما در بیشتر موارد داده ها در مسیر دیگری در دیاگرام جریان می یابند، حذف داده ها و همچنین، آپدیت هایی که رشته های با طول متغیر را گسترش می دهند منجر به منقضی شدن داده ها می شود که این خود منجر به DB-Slack می شود که در دیاگرام ۱ فلش بالایی که از حالت فعال خارج می شود آن را نشان می دهد.

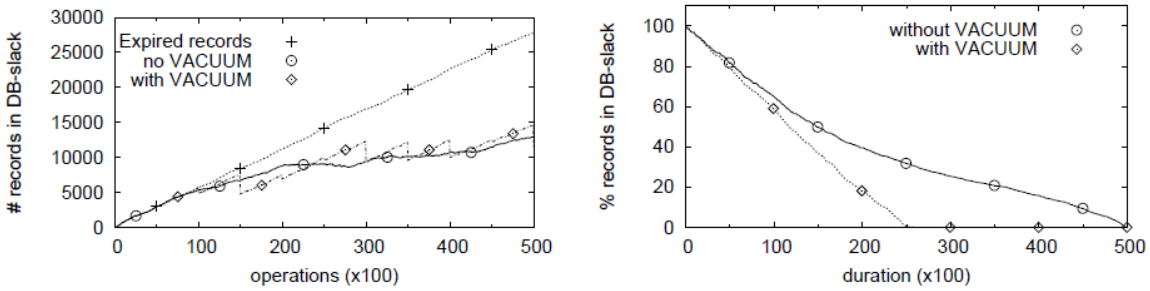
هنگامی که داده ها تبدیل به DB-Slack شدند بعدا به دو شرط می توانند کاملا از بین بروند: اول، درج در پایگاه که ممکن است DB-Slack را بازنویسی مجدد کند؛ دوم، پروسه Vacuum که رکورد ها را در فایل دوباره سازماندهی کند ممکن است DB-Slack را رونویسی کند. پروسه Vacuum ممکن است فضای تخصیص یاقته به فایل را به سیستم فایل بازگرداند و در نتیجه این کار مقداری DB-Slack به جای از بین رفتن داده ها می تواند به FS-Slack تبدیل شود. در واقع ما فهمیده ایم که Vacuum می تواند با کپی رکورد های فعال DB-Slack یا FS-Slack تولید نماید. بنابراین حالت ها در شکل ۱ متقابلا منحصر به فرد نیست، اقلام داده می تواند همزمان فعال و در FS-Slack باشند. سرانجام FS-Slack می تواند با یک عملیات معمولی از بین برود که در آن یک پروسه جدید (رونویسی یا صفر) سیستم را کاملا پاکسازی می کند.

[1] fragmented

[2] Vacuum



شکل ۱: یک نمودار حالت که جریان داده در زمان حیات خود را توصیف می کند. داده ها در حالت فعال حیات خود را آغاز می کنند. داده قبل از اینکه حذف و از بین برود اثرات خود را به صورت DB-Slack و FS-Slack به جای می گذارد.



شکل۲: میزان داده های قابل بازیابی برای سیستم های MySQL که از فرمت جدول InnoDB استفاده می کنند، زیر بار عملیات ترکیبی از حذف، درج و آپدیت: الف) DB-Slack با و بدون عملیات Vacuum. ب) اندازه گیری تجمعی طول عمر تاپل ها در DB-Slack.

آزمایش حجم کار

برای فهمیدن اثر یافته هایمان ما DB-Slack و FS-Slack را در سیستم های واقعی تحت عملیات شبیه سازی شده درج، حذف و آپدیت تاپل ها بررسی کردیم. ما یک ابزار قانونی اولیه برای تفکیک فایل های پایگاه داده و بازیابی DB-Slack و از تکنیک های قانونی سیستم فایل نیز برای بازیابی FS-Slack استفاده کرده ایم.

اگرچه تمام سیستم ها در خواص پایه مشخص شده در بالا مشترک هستند، ما بعضی تفاوت های جالب را با استفاده از حجم عملیات شبیه سازی شده فهمیدیم. PostgreSQL دارای بالاترین حجم باقی گذاشتن داده در ذخیره سازی جدول بود. تاپل ها هیچ وقت رونویسی نمی شوند به جز در مواقع فراخوانی Vacuum، که این از پیامدهای پایداری داده های به جا مانده در مدیر ذخیره سازی PostgreSQL است (بخش ۴,۱ را ببینید). دو پایگاه داده MySQL (که از فرمت جدول MyISAM استفاده می کند) و DB2 به طور معمول دارای پایین ترین میزان باقی گذاشتن داده در جدول ذخیره سازی با تنظیمات پیش فرض هستند. هر چند در هر دو مورد ما یافته ایم که ایندکس های دسته بندی شده می تواند به طور چشمگیر Slack داده را در جدول ذخیره سازی ایجاد کند. این بدان دلیل است که clustering استفاده مجدد از فضای آزاد در ذخیره سازی رکورد را تحمیل می کند، که این احتمال رونویسی داده در DB-Slack را کاهش می دهد. اضافه بر این، DB2 دارای دو پارامتر قابل تنظیم[۳] در نظر گرفته شده برای افزایش کارایی درج و آپدیت بوسیله رها کردن فضای خالی در صفحات و رها کردن تعدادی از صفحات آزاد در طول فراخوانی Vacuum است. اینها موثرا به عنوان دستگیره برای پایگاه داده به شمار می آیند که باعث افزایش عملکرد در هزینه های نگهداری نا خواسته داده در پایگاه داده می شود.

[۳] این پارامتر ها FREEPAGE و PCTFREE نامیده می شوند.

به عنوان یک مثال از نتایج تجربیاتمان، ما تاثیر Vacuum بر قابلیت بازیابی داده در MySQL را توصیف می کنیم که از فرمت جدول InnoDB استفاده می کند. ما فهمیدیم که Vacuum باعث افزایش داده در DB-Slack می شود. چون سازماندهی مجدد جدول بوسیله کپی ساده تمام تاپل های فعال به بخشی خالی از فضای جدول ذخیره سازی انجام می گیرد. شکل (۲- الف) نتایج اخیر را تشریح می کند، رشد DB-Slack در MySQL (InnoDB) را نشان می دهد. خط رکورد های منقضی شده در شکل بیشترین میزان داده های قابل بازیابی را نشان میدهد. در صورتی که دو خط دیگر شمار تاپل های قابل بازیابی منقضی شده با و بدون Vacuum را نشان می دهد. اگرچه Vacuum شمار تاپل های قابل بازیابی از DB-Slack را افزایش می دهد اما این خود باعث کاهش طول عمر مورد انتظار تاپل ها می شود همانطور که در شکل (۲- ب) نشان داده شده است.

۲,۲ تحلیل قانونی ایندکس ها

در اینجا دو منبع برای بازیابی قانونی از ایندکس های B-Tree وجود داردف اول، حذف از B-Tree ها منطقی است کلید های مرتب حذف شده رونویسی نمی شوند و می توانند در گره های داخلی باقی بمانند(بسیار شبیه جدول ذخیره سازی بالا). دوم، B-Tree ها یک پروسه استاندارد قطعی برای درج و حذف دارند بنابراین ممکن است از ساختار B-Tree ها اطلاعات جزئی در مورد ترتیب و توالی اضافه و حذف که منجر به وضعیت فعلی پایگاه داده می شود را استنباط کرد.

به عنوان یک مثال دو جدول با ایندکس را در نظر می گیریم که هر دو تحت عملیات درج ۱۰۰ عنصر قرار می گیرند، در جدول اول درج را بر اساس مرتب سازی صعودی و در جدول دوم درج را بر اساس مرتب سازی نزولی انجام می دهیم. اگرچه کلید مرتب سازی هر دو یکی استاما ساختار B-Tree ایجاد شده کاملا متفاوت خواهد بود که این نشان دهنده این است که تاریخچه توسط عملیات B-Tree حفظ می شود. به طور کلی بازرسی از ساختار درخت ممکن است به محقق این استنباط را بدهد که رکورد با کلید k_1 در جدول قبل از رکورد با کلید k_2 درج شده است. اینها اطلاعاتی در مورد رکوردها در پایگاه داده هاست و خارج از مدل داده است و نمی تواند از طریق رابط پرس و جوی پایگاه داده جمع آوری شود. مواردی وجود دارد که منظور از اتفاقات یک جنبه مهم و حیاتی برای آنالیز قانونی یا حسابرسی است.

درجه ای که ساختارهای داده ای اطلاعاتی را در مورد حالت های گذشته خود نشان می دهند در نظر گرفته شده است قبل از [38, 36]، اگرچه برای B-Tree ها در پایگاه داده ها در نظر گرفته شده است. تجزیه و تحلیل دقیق اطلاعات است که می تواند از ساختار B-Tree ها جمع آوری شود و این یک مسئله مهم را باز می کند. هر چند تاثیر عملی از این تجزیه و تحلیل ممکن است کوچک باشد. اول ما فرض می کنیم در یک پایگاه داده رابطه ای اطلاعات جمع آوری شده می تواند بوسیله اطلاعاتی که از مطالعه OIDS اختصاص یافته به تاپل ها یا آثار دیگر در پایگاه داده بدست آمده استنتاج شود. دوم ساختار B-Tree ها نشان می دهد که به احتمال زیاد مقادیر(شماره کلید در هر گره داخلی) به ترتیب کوچک می شوند و B-Tree ها معمولا high order هستند.

تحلیل تاریخچه[۴] از ساختار های ایندکس ممکن است بیشتر مربوط به جداول پایگاه های داده جاسازی شده[۵] باشد، مانند ایندکس های BerkeleyDB [5]. این جداول ایندکس اغلب برای داده های برنامه ها استفاده می شود و معمولا ساخته و به محیط های نا امنی که در آن وجود باقیمانده ها، داده ها و تاریخچه یک تهدید جدی است انتقال داده می شوند. ما به وضوح در بخش بعد به بررسی تکنیک هایی در مورد رفع این مشکل خواهیم پرداخت.

۲,۳ تحلیل قانونی لوگ تراکنش ها

پیش گزارش گیری از رایج ترین استراتژی های گزارش گیری در سیستم های امروزی است [43]. پس از هر آپدیت پایگاه داده یک رکورد گزارش از تصویر ماقبل و مابعد تغییر

[۴] History

[۵] embedded

داده ها تهیه می شود. بنابراین برای بازه های زمانی که گزارش تهیه شده است تمام حالت های قبلی پایگاه داده می تواند بازسازی شود و مقادیر زیادی داده می تواند بازیابی شود.

در حالی که جمع آوری داده ها از لوگ صریح و درست است، اما تعیین طول دوره احتباس داده بسیار مشکل است. لوگ تراکنش ها اغلب به وسیله لیست های حلقوی پیاده سازی می شود و رکوردهای لوگ به صورت پی در پی رونویسی می شوند. به مرور که لیست بزرگ می شود رکورد های قدیمی رونویسی شده و از بین می روند. زمانی که داده ها در لوگ تراکنش ها باقی می مانند به میزان فضایی که به لوگ تخصیص یافته تعداد عملیات آپدیت، فضایی که برای هر آپدیت لازم است و نقاط چک متناوب بستگی دارد. یک شرکت می تواند چرخه لوگ را در چند روز کامل کند که به طور موثر داده ها را در سیستم نگهداری می کند. اما سیستم های دیگر ممکن است لوگ تاریخچه های بزرگ که به طور نامحدود کار می کنند تولید کنند. در بخش بعد ما تکنیک هایی را در مورد حذف رکورد های لوگی که بیشتر از این برای لغو عملیات یا بازیابی مورد نیاز نیستند بحث می کنیم.

۳ طراحی سیستم های شخصی

مشکلی که در تحلیل قانونی نشان داده شد این است که رابط کاربری استاندارد پایگاه داده (به عنوان مثال SQL) به طور قابل اعتماد محتوای واقعی پایگاه داده را نشان نمی دهد. تاپل های حذف شده که در نتایج جستجو ظاهر نمی شوند اما در پایگاه داده وجود دارند، تاپل ها در مدل های داده در نظر گرفته شده "عمر" یا نظم در ایجاد شدن ندارند هنوز یک دستور را می توان از نمایش فیزیکی بازیابی کرد. قطع ارتباط بین رابط کاربری و سیستم واقعی یک مشکل جدی برای صاحبان داده است چرا که یک شمای غلط و نا صحیح از ذخایر داده را به کاربر نشان می دهد.

برای حل این مشکل پیشنهاد می کنیم که سیستم شفاف[۶] باشد. به طوری که داده های ذخیره شده به درستی نشان داده شوند. به طور مطلوب داده های حذف شده توسط کاربران باید به طور کامل از بین بروند و همچنین تمام کپی های موجود از آن داده حذف شود. اگر صرف نظر از حذف داده ها برای یک هدف مشروع و قانونی نگهداری شوند، کاربران باید از مرزهای طول عمر داده ها به صورت روشن و دقیق باخبر باشند و آنها بتوانند این مرزها را با استفاده از پارامتر های سیستمی تغییر دهند. شفافیت به کاربران اجازه می دهد تا از سیاست های حفظ حریم خصوصی که سیستم برای آنها فراهم می کند آگاه باشند.

توجه داشته باشید که تعریف ما از شفافیت سیستم در تضاد با دلایل مشروع و قانونی برای حفظ داده ها پش از حذف نیست. اینها شامل versioning پایگاه داده، که حالت های گذشته پایگاه داده حفظ و می تواند پرس و جو شود، مکانیزم بازیابی پایگاه داده، که داده های حذف شده را در لوگ تراکنش ها برای حفظ اتمیزه بودن و پایداری نگهداری می کند و پشتیبان ها که برای بازرسی و بازیابی از رسانه زمانی که دچار مشکل می شود هستند. اول از همه رمانی که پشتیبان گیری و آرشیو کردن انجام می گیرد، این خیلی مهم است که تضمین شود فقط داده های معتبر مورد نظر ذخیره می شوند نه بقایای ناخواسته از حالت های گذشته، دوم اینکه اگر به طور نا مناسب به کار برده شود ذخایر داده های مورد نظر باید از طریق یک رابط کاربری مشروع و تنظیم شده در معرض دید قرار داده شوند. در واقع یک راه سر راست از آرشیو کردن شفاف داده این است که داده ها در نسخه هایی بر حسب تاریخ از پایگاه داده ذخیره شده و به طور ساده و درست از طریق یک رابط کاربری قابل پرس و جو باشند (این از اهداف بخش ۴ است).

همیشه مشکلاتی نیز وجود دارد به هر حال versioning همیشه مطلوب نیست. در این مورد پایگاه داده باید داده را به طور ایمن حذف کند، ایندکس هایی را فراهم کند که وابسته به تاریخ باشند، و شفافیت را در برنامه هایی که در

[۶] Transparent

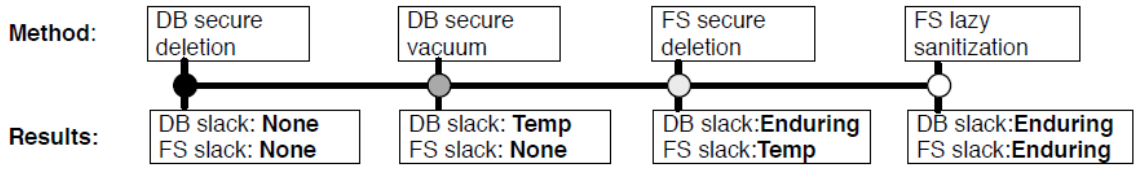
پایگاه داده ساخته می شوند فراهم کند. ما این چالش های تکنیکی را در بخش بعد بررسی می کنیم.

۳,۱ حذف ایمن از پایگاه داده

جامعه ایمن، حذف ایمن گزارش های پشتیبان گیری شده را مورد مطالعه قرار می دهد [9]، روش های سنتی و versioning سیستم فایل [6, 42] و برای مدیر ذخیره پست الکترونیکی یک امکان منقضی کردن پیام بر اساس زمان را فراهم می کند [41]. در اینجا دو تکنیک پایه و اساسی برای حذف فیزیکی داده وجود دارد: تخریب فیزیکی با رونویسی داده و رمزگذاری با استفاده از یک کلید در دسترس. امنیت رونویسی برای حذف داده ها اولین بار توسط گاتمن [31] مورد بررسی قرار گرفت. او استدلال می کند که داده های رونویسی شده می تواند از دیسک های ذخیره سازی بازیابی شوند، اگرچه این موضوع با توجه به تراکم دیسک های مدرن به طور فزاینده ای بعید است [6, 26]. با این حال رونویسی هر بایت حذف شده با صفر ممکن است یک کار سنگین عملیاتی برای بلوک های بزرگی از داده ها به ما تحمیل می کند. یک جایگزین هوشمندانه برای این کار اولین بار توسط بونه و لیتون پیشنهاد شد [9]. ذخیره داده ها به صورت رمزگذاری شده و انجام کار روی کلید رمزگذاری (حذف یا رونویسی) مادامیکه رونویسی کلید مناسب باشد، رمزگذاری بایت به بایت داده باید با حذف بایت به بایت آن مطابق باشد. برای مثال در جدول ذخیره سازی، در حذف ایمن تاپل ها ممکن است برای هر تاپل یک کلید نیاز باشد. سربار مدیریت کلید و همچنین سربار رمزگشایی/رمزگذاری برای هر عمل خواندن/نوشتن به احتمال زیاد این استراتژی را نامناسب می سازد.

پایگاه داده و مسائل سیستم فایل

اضافه بر این که چگونه حذف ایمن را انجام می دهیم، ما باید بخشی از سیستم فایل را که در برابر پیاده سازی حذف مسئول است را در نظر بگیریم. شکل ۳ طیفی از راه حل های ممکن برای حذف ایمن را خلاصه سازی و نشان می دهد.



شکل ۳: طیف راه حل های قابلیت بازیابی داده ها: از پایگاه داده به سیستم فایل

انتخاب به بده بستان های مختلفی در عملکرد امنیت و پیچیدگی و مهندسی راه حل بستگی دارد. در یک انتهای طیف حذف ایمن و آپدیت می تواند توسط پایگاه داده پیاده سازی شود. در این مورد DB-Slack و FS-Slack به طور کامل از بین رفته چرا که داده بلافاصله از بین می رود. برای بدست آوردن یک حد بالای عملکردی از حذف ایمن ما یک حذف ایمن ساده شامل یک آپدیت (رونویسی هر خصوصیت یک رکورد با صفر) پس از حذف را آزمایش کردیم. ما از MySQL (MyISAM) با حذف تصادفی از یک جدول ایندکس از خصوصیت های با اندازه ثابت استفاده کردیم. شبیه سازی های حذف ایمن نرخ اجرای عملیات حذف را به حدود نصف کاهش داد، از ۱۳۱۷۵ در هر ثانیه به ۶۹۹۵ در هر ثانیه. این پیاده سازی سطح کاربر ساده ازحالت بهینه بسیار دور است. این نشان می دهد که می توان عملکرد معقول را با یکپارچه سازی حذف ایمن به سیستم پایگاه داده بدست آورد. همچنین زمانی که عملیات حذف بخشی از یک حجم کار عظیم پرس و جوی خواندن و نوشتن است ممکن است سربار مشاهده شده قابل قبول نباشد.

اگر DB-Slack موقت قابل تحمل باشد، حذف ایمن ضعیف با پایگاه داده برای ارائه عملکرد بهتر انجام خواهد شد. اگرچه برنامه های کمی ممکن است روش حذف ضعیف (غیر ایمن) را بخواهند اما آنها حذف ایمن را با روش

Vacuum ترکیب می کنند. مثلا یک پروسه اجرای Vacuum ایمن که رکورد ها را در صفحه دوباره سازماندهی می کند، هرگونه داده منقضی شده را از بین برده و داده های Slack را قبل از اینکه فضا به سیستم فایل برگردانده شود را پاک می کند پس بدینوسیله اجتناب از FS-Slack نیز به خوبی انجام شده است.

اعمل تغییرات به سیستم فایل منجر به ارائه عملکرد بهتر می شود، اما منجر به از دست رفتن امنیت می شود. بور و همکارانش روش حذف ایمن سنکرون را در سیستم فایل هایی که به طرز غیر قابل قبولی کند هستند را بدست آوردند، اما یک نمونه تحقیقی از سیستم فایل ext2 [15] که با حذف ایمن غیر سنکرون با رونویسی کار می کرد پیاده سازی شد. استفاده از این سیستم فایل باید FS-Slack را پس از یک دوره زمانی کوتاه از بین ببرد اما نباید DB-Slack را پاک کند. در ضمن مزیت ارتقای سیستم فایل یک کلیت است: این کار باعث اجتناب از تغییر هر یک از سیستم های پایگاه داده می شود (و هر برنامه ای که داده های ساخت یافته ذخیره می کند) که این خود از شفافیت مورد نیاز تبعیت می کند.

بدبختانه ما از هر سیستم فایلی که به طور رایج استفاده و حذف ایمن را فراهم می کند آگاه نیستیم. بنابراین راه حل نهایی پاکسازی سیستم فایل با صلاحدید کاربر یا مدیر پایگاه داده است. برنامه ها و ابزارهایی برای پاکسازی دیسک وجود دارند که می توانند به صورت دوره ای برای پاک کردن داده از فضای اختصاص نیافته سیستم فایل تنظیم شوند. این برنامه ها اغلب فاقد امتیازات لازم برای فراهم ساختن حذف ایمن به طور کامل هستند. اغلب آنها سعی می کنند تا فضای آزاد را با یک فایل حجیم پر کنند که منجر به رونویسی تمام بایت ها می شود. در این روش ممکن است بقایای ابر داده[۷] فایل قابل بازیابی باشد.

با در نظر گرفتن این ملاحظات دیدگاه ما این است که مدیر ذخیره پایگاه داده باید حذف ایمن را پشتیبانی کند. ما معتقدیم که حذف ایمن می تواند با رونویسی و بدون یک سربار عملکردی غیر قابل قبول قابل پیاده سازی باشد. به خصوص اگر یک تاخیر زمانی کوچک مجاز باشد.

پاکسازی لوگ تراکنش ها

این سخت نیست که بخش هایی از لوگ تراکنش ها که هرگز توسط مدیر تراکنش ها برای بازیابی یا لغو استفاده نمی شود شناسایی شده و بتواند آزادانه حذف شود. که این معمولا لوگ ثبت شده پس از چک پوینت ما قبل آخر است. در برخی زمینه ها حذف به موقع این داده ها ممکن است مطلوب باشد. رمزگذاری با کلید در دسترس در حالی که احتمالا برای ذخیره سازی جدول نا مناسب است، یک روش امید بخش برای حذف ایمن در لوگ تراکنش است. رمزگذاری به طور ویژه در این مورد مناسب است چون رکورد های لوگ فقط یک بار نوشته می شوند که فقط به یک عمل رمزگذاری نیاز دارد. اگر با شکست مواجه شدن سیستم یا لغو تراکنش پیش نیاید رمزگذاری ممکن است هیچ وقت نیاز نباشد. کلید ها می توانند در جدول تراکنش ها ذخیره حذف شوند. ما فرض می کنیم که سربار رمزگذاری/رمزگشایی رکورد های لوگ در طول عملیات معمولی پردازش تراکنش کوچک باشد و حذف ایمن ممکن و شدنی باشد.

۳,۲ ایندکس های مستقل از تاریخ

همان طور که در بالا اشاره شد، ایندکس های پایگاه داده می توانند اطلاعات در مورد وقایای گذشته را از طریق ساختار های خود و از طریق نمایش داخلی نشان می دهند. تعریف رسمی اصطلاح "ضد تداوم" در [36 ,38] توسعه داده شده است. یک درخت ۳-۲ "فراموشکار" در [36] پیشنهاد شده است که شکل آن نشان می دهد که هیچ چیز در مورد عملیات گذشته به درخت اعمال نمی شود. ساختار های داده "مستقل از تاریخ" طراحی شده است [38] که از آشکار سازی های حاصل از شکل و از نمایش حافظه از ساختار داده جلوگیری می کند.

یک تکنیک برای یک ایندکس B-Tree ایمن این است که بعد از هر بروز رسانی یک فرم متعارف اجرا شود. این به طور

[۷] Metadata

جلوگیری کننده ای گران است. یک تکنیک دیگر تصادفی کردن عملیات یا دوباره متعادل کردن محلی است، که می تواند آشکار سازی ها را بپوشاند، طراحی ایندکس های B-Tree که اطلاعات تاریخی آشکار را بدون فدا کردن کارایی محدود کند یک چالش رایج است.

۳,۳ راهنمایی برای متخصصان

در یک عبارت کوتاه، یک طراحی مجدد زمان بر و هزینه بر از پایگاه داده و سیستم فایل داخلی یک راه حل معقول و منطقی برای مقابله با تهدید ها علیه امنیت سیستم پایگاه داده نیست. بنابراین ما روش هایی ساده را پیشنهاد می دهیم که نگهداری داده های ناخواسته در پایگاه داده را کاهش خواهد داد:

- لوگ تراکنش باید یک فایل حلقوی باشد. بیشترین فضای اختصاص داده شده به فایل لوگ یک پارامتر قابل تنظیم است که باید در اتباط با نرخ نقطه بازگردانی گرفتن تنظیم شود تا مطمئن شویم که لوگ با یک فرکانس قابل قبول بچرخد.
- برای صفات خاصه حساس (مانند: کارت اعتباری و شماره امنیت اجتماعی) ممکن است شبیه سازی حذف ایمن در سطح کاربر با ارزش باشد همانطور که در بخش ۳,۱ به آن پرداخته شد. از رونویسی مقدار داده های منقضی شده با Null یا صفر باید اجتناب شود، از آنجا که در بسیاری از سیستم ها (DB2, InnoDB, MySQL, PostgreSQL) این داده ها را به صورت قابل بازیابی رها می کنند.
- در بیشتر سیستم ها vacuum داده های باقیمانده در DB-Slack را پاک خواهد کرد (InnoDB و DB2 استثنا هستند) و FS-Slack را به جای می گذارد. پس از vacuum باید بلافاصله پاکسازی سیستم فایل اجرا شود.

۳,۴ ترویج شفافیت در برنامه های کاربردی پایگاه داده

حذف غیر ایمن و باقیماندن داده های نا خواسته یک مشکل سیستمی است. که در سیستم فایل ها [6]، حافظه فرار [20, 19] و اکنون در ذخایر پایگاه داده مورد مطالعه قرار گرفته است. واقعیت این است که هر وقت فضا برای ندیریت توسط یک بخش سطح بالا از سیستم اختصاص یابد، حذف ممکن است به صورت غیر ایمن انجام شود. به خصوص این الگو ممکن است تکرار شود هر وقت شمای پایگاه داده شامل انواع داده BLOB یا CLOB باشند. در آن صورت پایگاه داده بلوک بزرگی از بایت ها را اختصاص می دهد و این موجب می شود که نتوان حذف ایمن داده های ذخیره شده در بلوک را انجام داد وقتی که بدون یک دستورالعمل صریح از برنامه کاربردی منقضی شود. BLOB/CLOB به طور رایج در ذخیره سازی اشیاء پیچیده استفاده می شود، داده های چند رسانه ای و داده های XML و باقیماندن این داده ها ممکن است به وجود آورنده یک تهدید جدی باشد. تکنیک های حذف ایمن که در بالا توصیف شد باید در سطح کاربر برای طراحان برنامه های سطح پایین پیاده سازی و در معرض استفاده قرار داده شود تا باقیمانده های داده به صورت ایمن حذف شوند.

۴ طراحی سیستم های پاسخگو

در بخش قبلی ما در مورد چالش هایی درباره طراحی سیستم های پایگاه داده که به طور قابل اعتماد حالت واقعی ذخیره داده را به کاربر برای تنظیم رفتار سیستم با یک خط مشی امنیتی مطلوب نشان می دهد بحث کردیم. در این بخش ما به مشکل مکمل طراحی سیستم های پایگاه داده پاسخگو می پردازیم. این سیستم ها نیازمند دو قابلیت کلیدی هستند. اول داده های تاریخی کامل باید به طور مناسب برای فراهم آوردن پاسخگویی به پرسش ها گردآوری و یکپارچه شود. دوم، داده های تاریخی باید محافظت شده باشد، که نیاز به گسترش سیاست های فعلی کنترل دسترسی و پشتیبانی از عملیات پاکسازی و ویرایش به صورتی که در زیر توصیف شده دارد. قبل از بحث در مورد این چالش ها ما قابلیت های سیستم های موجود را بازبینی می کنیم.

۴,۱ قابلیت های موجود

سیستم های پایگاه داده قدیمی برای دستیابی مناسب به حالت فعلی پایگاه داده طراحی شدند (اغلب نمایش لحظه ای نامیده می شد). با استفاده از لوگ تراکنش حالت های گذشته پایگاه داده می تواند بازیابی شود. برای مثال تعدادی از سیستم ها "بازیابی نقطه در زمان" را پشتیبانی می کنند، که به طور نمونه تراکنش ها را از یک نسخه پشتیبان دوباره راه اندازی می کند [39]، اما همچنین ممکن است از نسخه فعلی به عقب براند. در هر صورت دوباره راه اندازی یک حالت گذشته از طریق تکرار تراکنش ها یک عملیات گران است که باید قبل از پرس و جوی حالت گذشته ای که باید اجرا شود کامل شود. اگر ما مایل به تجزیه و تحلیل باشیم، برای مثال تکامل یک تاپل تک از میان حالت های پایگاه داده، این ویژگی ها ناکافی هستند. پایگاه های داده قدیمی که ورود عملیات را پشتیبانی می کنند، رکورد ها را از عملیات مدیریتی، پرس وجو های تایید شده، اتصال های راه دور و غیره محافظت می کند.

حفاظت از سوابق در پایگاه های داده توجه قابل ملاحظه ای را از جامعه تحقیق و پژوهشی می طلبد و با نام های بسیاری معروف است: پایگاه های داده مقاوم، بایگانی کردن پایگاه های داده [11]، versioning پایگاه داده [53]، زمان تراکنش پایگاه های داده [34, 35]. سیستم Postgres در ابتدا برای پشتیبانی از versioning طراحی شد، که در آن تاپل های منقضی شده باقی مانده و مهر زمانی بر اساس زمان تایید[۸] تغییر تراکنش می خورند. یک عمل حذف هیچ گاه داده را از ابین نمی برد و اعمال به روز رسانی به آیتم های داده یک ورژن جدید ایجاد می کند. ممکن است که به طرح پرس و جو ها در هر نقطه از زمان گذشته بپردازد. پشتیبانی از versioning کامل در سیستم های تجاری به طور کسترده پیاده سازی نشده است. قابلیت های versioning در سیستم PostgreSQL از ورژن ۶/۳ به بعد ظاهرا به دلایل عملکردی [1] (ارائه در مارس ۱۹۹۸) حذف شد. تکنیک های پایگاه داده موقتی همچنین به طور نزدیک مرتبط است [40]، و شماری از مدل های داده ای و زبان های پرس و جو ارائه شده اند. تعدادی از محققین نمونه های آزمایشی در تلاش برای ساختن پایگاه های داده موقتی یا زمان تراکنش، در بالای سیستم های موجود مانند MySQL [51]، BerkeleyDB [50] و SQL Server [35] هستند. TSQL [52] یکی از ملحقات معروف SQL است که قابلیت های موقتی[۹] را فراهم می کند.

هر چند که کار های زیادی در این زمینه وجود دارد، شماری از چالش ها در مورد پاسخگویی وجود دارد. ما روی چالش های ساخت یک پایگاه داده با قابلیت versioning تمرکز نمی کنیم، ما فرض می کنیم که چنین سیستمی موجود است شاید پس از کار های اخیر مدل سازی شود [51, 35].

۴,۲ یکپارچه سازی و پرس و جوی داده های سوابق

پاسخگویی به پرس و جو ها اغلب شامل تجزیه و تحلیل دنباله یا سری زمانی از مقادیر در یک پایگاه داده است. زمانی که کاربرانی عملیاتی را انجام می دهند [18]. برای پشتیبانی از پرس و جو ها ما پیشنهاد می کنیم که داده های پایه همراه با سوابق عملیات کاربر شامل: ایجاد، به روز رسانی و یا انتخاب رکورد های داده یکپارچه و تجمیع شود. اکنون تعدادی از این داده ها در لوگ تراکنش ها یا عملیات ثبت شده و به طور نمونه بخشی از یک ذخیره داده نگارش بافته یا قدیمی نیست. ما پیشنهاد می کنیم که یک مدل داده یکپارچه سازی شده با عدم تفاوت منطقی بین داده های پایه و سوابق عملیات داشته باشیم. به طور فیزیکی پاسخگویی ابرداده ممکن است به طور مجزا از داده های پایه ذخیره شود، اما باید توسط سیستم پایگاه داده مدیریت شود.

این می تواند به عنوان یک سیستم پایگاه داده نگارشی که اصل پاسخگویی را فراهم می کند استدلال شود. اصل پاسخگویی پایگاه داده مربوط به ردیابی و ثبت مشاء داده ها و حرکت انها در پایگاه داده است. در اینجا یک کار

[۸] commit

[۹] Temporal capabilities

اساسی در مورد منشا داده ها وجود دارد [13 ,12 ,21]، شامل دو نمونه تحقیقی آزمایشی اخیر [10 ,3]. به هر حال تمرکز ما روی حفظ اصل امنیت در یک پایگاه داده است که این در تضاد با مدیریت منشاء ذخیره های داده یا پایگاه داده یکپارچه که به طور معمول مورد مطالعه قرار گرفته است است. پاسخگویی به پرس و جو ها نیازمند تاکید به این نکته مهم است که چه کسی چه کاری را انجام داده و آنها در چه زمانی انجام شده اند.

مدیریت پاسخگویی داده ها به بازرسان اجازه خواهد داد که الگو های دستیابی و فعالیت کاربران را تجزیه و تحلیل کنند. اما صرفا پرس و جو های "as-of" را مطرح می کند، که پاسخ های محکم به عنوان لحظه ای در زمان گذشته تولید می کند، کافی نخواهد بود. قابلیت های جدید زبان لازم خواهد بود تا پاسخگویی به پرس و جو ها را بهبود بخشد. شماری از زبان های پرس و جو در زمینه پایگاه های داده موقتی توسعه داده شده اند [40]. آنها مقایسه در سراسر نسخه ها را ممکن می سازند اما ممکن است نیاز باشد برای یکپارچه سازی پاسخگویی داده گسترش یابند.

۴,۳ حفاظت از تاریخچه

محافظت از تاریخچه به معنای تعریف و اعمال حقوق دسترسی به داده های باقیمانده سوابق است. حفاظت از داده های سوابق از خود داده ها بسیار مهمتر است، بنا به تعریف برای مدت زمان طولانی ذخیره می شود. کنترل دستیابی در پایگاه های داده قدیمی فقط نمایش لحظه ای را تنظیم می کند. هنگامی که به تاپل های نگارش یافته اعمال شد (به طور معمول در جداول رابطه ای با فیلد مهر زمانی نشان داده شده) کنترل دستیابی SQL نامناسب است چرا که سیسات های سطح-ردیف[۱۰] را پشتیبانی نمی کند. هنگامی که مدل های کنترل دستیابی موقتی [8, 4, 7] ارائه شد، آنها به تغییر حق دسترسی در طول زمان تاکید می کنند که از تعیین حقوق دسترسی پایدار با مجموعه های داده ای متغیر با زمان متفاوت است. طراحی یک مدل کنترل دسترسی برای پشتیبانی از سیاست های توصیفی از داده های تاریخی به نظر می رسد یک مشکل جدی باشد. قوانین کنترل دسترسی باید قادر به تعیین شرایط رسا و گویا در مورد زمان و عملیات پایگاه های داده و همچنین اشیاء داده ای باشد. لوگ بازرسی و تراکنش ها توسط مکانیزم های صدور مجوز پایگاه های داده کلاسیک قابل محافظت نیست. بیشتر سیستم ها فقط کنترل دستیابی های سخت را به لوگ ها اعمل می کنند که توسط سیستم فایل پیاده سازی شده است که در آن مجوز برای دسترسی به لوگ نیازمند دسترسی کامل است.

ویرایش و پاکسازی در یک پایگاه داده با قابلیت versioning

حتی در یک سیستم versioning که حذف صریح فراهم نمی شود مقادیر داده ای حساسی هستند که باید به طور کامل حذف شوند. برای مثال یک شرکت ممکن است بخواهد سوابق گذشته حساب های مشتریانش را نگهداری کند و در بعضی مواقع بخواهد تمام رکورد های شماره حساب های مشتریانش را حذف کند، ما معتقدیم که یک چنین عملی باید یک قابلیت اساسی سیستم پایگاه داده باشد که سابقه را نگه دارد، و باید به عنوان یک عملیات reduction پیاده سازی شود. Reduction نیازمند تعیین هویت مناسب تمام کپی های رکورد های ذخیره شده از مقادیر داده ای است که باید به طور ایمن حذف و با یک مقدار خاص رونویسی شوند تا نشان دهد که داده ها کاملا پاک شده اند. این امر ممکن است با یک عملیات پاکسازی مقایسه شود که تمام آثار و شواهد رکورد ها را حذف می کند. توجه داشته باشید که در صورتی که به عنوان نمونه آپدیت ها در یک version پایگاه داده فقط نسخه فعلی را تغییر می دهند. این عملیات ممتاز سوابق را تغییر می دهند بنابراین در اینجا چالش هایی برای سازگاری پرس و جو در این مدل وجود دارد: اگر reduction تاریخی ممکن باشد، پرس و جو هایی در حالت های گذشته پایگاه داده باید جواب هایی شامل منابع و مراجع برای مقادیر نوشته شده تولید کند که هنوز برای کابران معنی دار است.

[۱۰] Row-level

مدل کنترل دسترسی مطلوب سیاست های توصیف درست را در این مدل داده گسترش یافته فراهم می کند و امتیازات جدید را در خود جای می دهد. مکانیزم های اجرای چنین سیاست هایی به احتمال زیاد بر دو اقدام کنترل دسترسی فعال تکیه می کند (که در آن یک ناظر کنترل دستیابی مرسوم دسترسی را تنظیم می کند) همچنین اصطلاحا اجرای کنترل دسترسی منفعل (که در آن رمزنویسی برای تنظیم دسترسی استفاده می شود). کنترل دسترسی منفعل به نظر می رسد برای اجرای کارآمد پاکسازی و ویرایش مناسب باشد.

۵ نتیجه گیری

ما معیار های طراحی و چالش های تکنیکی در ساخت یک سیستم پایگاه داده که "حافظه تاریخی" می تواند به صورت سالم و دقیق مدیریت شود را توصیف کردیم. در تنظیماتی که امنیت حیاتی است پایگاه های داده باید داده های منقضی شده را در یک وضعیت زمانی حذف و سابقه فعالیت ها را از بین ببرند. در تنظیماتی که پاسخگویی در اولویت است یک سیستم پایگاه داده باید به کاربران اجازه دهد تا به طور موثر سیستم ها را نظارت کرده و به صورت ایمن از داده های حساس که از نظارت نتیجه شده اند حفاظت کنند. ما معتقدیم که مدیریت ایمن تاریخچه رشد خواهد کرد تا به یک ویژگی مهم از سیستم هایی که به طور فزاینده ای قادر به حفظ تمام حالت های گذشته هستند تبدیل شود و نگرانی در مورد امنیت نیازمند کنترل پاسخگویی است.

6 REFERENCES

[1] PostgreSQL release notes, v 6.3. Available at: www.postgresql.org/docs/current/static/release-6-3.html, March 1998.

[2] Serge Abiteboul, Rakesh Agrawal, Phil Bernstein, and et. al. The lowell database research self-assessment. *Commun. ACM*, 48(5):111–118, 2005.

[3] Parag Agrawal, Omar Benjelloun, Anish Das Sarma, Chris Hayworth, Shubha U. Nabar, Tomoe Sugihara, and Jennifer Widom. Trio: A system for data, uncertainty, and lineage. In *VLDB*, pages 1151–1154 , 2006.

[4] Vijayalakshmi Atluri and Avigdor Gal. An authorization model for temporal and derived data: securing information portals. *ACM Trans. Inf. Syst. Secur.*, 5(1):62–94, 2002.

[5] Berkeley db xml. Available at www.sleepycat.com.

[6] Steven Bauer and Nissanka B. Priyantha. Secure data deletion for linux file systems. In *Procedings of the 10th USENIX Security Symposium*, pages 153–164 , 2001.

[7] Elisa Bertino, Claudio Bettini, Elena Ferrari, and Pierangela Samarati. A temporal access control mechanism for database systems. *IEEE Transactions on Knowledge and Data Engineering*, 8(1):67–80, 1996.

[8] Elisa Bertino, Claudio Bettini, and Pierangela Samarati. A temporal authorization model. In *CCS '94: Proceedings of the 2nd ACM Conference on Computer and communications security*, pages 126–135, New York, NY, USA, 1994. ACM Press.

[9] Dan Boneh and Richard J. Lipton. A revocable backup system. In *USENIX Security Symposium*, pages 91–96, 1996.

[10] Peter Buneman, Adriane Chapman, and James Cheney. Provenance management in curated databases. In *ACM SIGMOD Conference on Management of Data*, pages 539–550, New York, NY, USA, 2006. ACM Press.

[11] Peter Buneman, Sanjeev Khanna, Keishi Tajima, and Wang Chiew Tan. Archiving scientific data. In *SIGMOD Conference*, 2002.

[12] Peter Buneman, Sanjeev Khanna, and Wang Chiew Tan. Why and where: A characterization of data provenance. In *ICDT*, pages 316–330, 2001.

[13] Peter Buneman, Sanjeev Khanna, and Wang Chiew Tan. On propagation of deletions and annotations through views. In *PODS '02*, pages 150–158, 2002.

[14] Simon Byers. Scalable Exploitation of, and Responses to Information Leakage Through Hidden Data in Published Documents, April 2003.

[15] R. Card, T. Tso, and S. Tweedie. Design and implementation of the second extended filesystem. In *Proc. Dutch International Symposium on Linux*, 2004.

[16] Brian Carrier. Sleuth toolkit / Autopsy forensic browser. Available at www.sleuthkit.org.

[17] Brian Carrier. *File System Forensic Analysis*. Addison-Wesley Professional, 2005.

[18] Eoghan Casey. *Digital Evidence and Computer Crime*. Elsevier, 2nd edition, 2004.

[19] Jim Chow, Ben Pfaff, Tal Garfinkel, Kevin Christopher, and Mendel Rosenblum. Understanding Data Lifetime via Whole System Simulation. In *Proc. USENIX Security Symposium*, August 2004.

[20] Jim Chow, Ben Pfaff, Tal Garfinkel, and Mendel Rosenblum. Shredding Your Garbage: Reducing Data Lifetime Through Secure Deallocation. In *Proc. USENIX Security Symposium*, August 2005.

[21] Yingwei Cui and Jennifer Widom. Practical lineage tracing in data warehouses. In *International Conference on Data Engineering*, pages 367–378, 2000.

[22] Encase forensic. Available at www.guidancesoftware.com.

[23] European union directive on privacy and electronic communications. register.consilium.eu.int/pdf/en/02/st03/03636en2.pdf.

[24] Ron Edmonds. Justice department hid parts of report criticizing diversity effort. Associated Press/USA Today, October 2003.

[25] U.S. Family Educational Rights and Privacy Act (FERPA). www.ed.gov/offices/OII/fpco/ferpa.

[26] Simson L. Garfinkel. *Design Principles and Patterns for Computer Systems That Are Simultaneously Secure and Usable*. PhD thesis, M.I.T., 2005.

[27] Simson L. Garfinkel and Abhi Shelat. Remembrance of data passed: A study of disk sanitization practices. *IEEE Security and Privacy*, Jan/Feb 2003.

[28] Tal Garfinkel, Ben Pfaff, Jim Chow, and Mendel Rosenblum. Data Lifetime is a Systems Problem. In *Proc. ACM SIGOPS European Workshop*, September 2004.

[29] Jim Gray. Database operating systems: Storage and transactions. Invited Talk, SIGMOD, 2006.

[30] Tim Grieve. The decline and fall of the enron empire. Salon Magazine, October 2003.

[31] Peter Gutmann. Secure Deletion of Data from Magnetic and Solid-State Memory. In *Proc. USENIX Security Symposium*, July 1996.

[32] U.S. health insurance portability and accountability act (HIPAA). www.hhs.gov/ocr/hipaa.

[33] Bryan Klimt and Yiming Yang. Introducing the Enron Corpus. In *in Proc. Conference on Email and Anti-Spam (CEAS)*, July 2004.

[34] David Lomet, Roger Barga, Mohamed Mokbel, German Shegalov, Rui Wang, and Yunyue Zhu. Immortal db: Transaction time support for sql server. In *SIGMOD Conference*, 2005.

[35] David Lomet, Roger Barga, Mohamed Mokbel, German Shegalov, Rui Wang, and Yunyue Zhu. Transaction time support inside a database engine. In *International Conference on Data Engineering*, 2006.

[36] Daniele Micciancio. Oblivious data structures: applications to cryptography. In *Symposium on Theory of Computing*, pages 456–464, 1997.

[37] Gerome Miklau, Patrick Stahlberg, and Brian Levine. Threats to privacy in the forensic analysis of database systems. Technical report, University of Massachusetts Amherst, 2006 *Submitted for Publication*.

[38] M. Naor and V. Teague. Anti-persistence: History Independent Data Structures. In *Proc. Symposium Theory of Computing*, May 2001.

[39] Oracle flashback technology. Available at www.oracle.com/technology/deploy/availability/ htdocs/Flashback Overview.htm, 2006.

[40] Gultekin Ozsoyoglu¨ and Richard T. Snodgrass. Temporal and real-time

databases: A survey. *IEEE Trans. Knowl. Data Eng.*, 7(4):513–532, 1995.

[41] Radia Perlman. The ephemerizer: Making data disappear. Technical Report TR-2005-140, Sun Microsystems, 2005.

[42] Z. Peterson, R. Burns, J. Herring, A. Stubblefield, and A. Rubin. Secure Deletion for a Versioning File System. In *Proc. File And Storage Technologies (FAST)*, pages 143–154, December 2005.

[43] R. Ramakrishnan and J. Gehrke. *Database Management Systems*. McGraw-Hill, 2000.

[44] Jeffrey Rosen. *The Unwanted Gaze: The destruction of privacy in America*. Random House, 2000.

[45] James M. Rosenbaum. In defense of the delete key. *The Green Bag*, 3, 2000.

[46] Sqlite. Available at www.sqlite.org.

[47] J. Shetty and Jafar Adibi. The enron email dataset database schema and brief statistical report. Technical report, Information Sciences Institute, 2004.

[48] Avi Silberschatz, Michael Stonebraker, and Jeff Ullman. Database systems: achievements and opportunities. *Commun. ACM*, 34(10):110–120, 1991.

[49] Avi Silberschatz, Mike Stonebraker, and Jeff Ullman. Database research: achievements and opportunities into the 1st century. *SIGMOD Rec.*, 25(1):52–63, 1996.

[50] Richard T. Snodgrass and Christian S. Collberg. The τ-BerkeleyDB temporal subsystem. Available at www.cs.arizona.edu/tau/tbdb/.

[51] Richard T. Snodgrass and Christian S. Collberg. The τ-MySQL transaction time support. Available at www.cs.arizona.edu/tau/tmysql.

[52] Richard Thomas Snodgrass. *The TSQL2 Temporal Query Language*. Kluwer Academic Publishers, Norwell, MA, USA, 1995.

[53] Michael Stonebraker and Lawrence A. Rowe. The design of postgres. In *SIGMOD Conference*, pages 340–355, 1986.

Source: http://people.cs.umass.edu/~miklau/pubs/cidr2007/miklau07securing.pdf