بسمه تعالی

سازمان فناوری اطلاعات ایران

معاونت امنیت فضای تولید و تبادل اطلاعات

مرکز ماهر

جرم‌شناسی در پایگاه‌داده MsSQL

پیشگفتار

رشد روز افزون حجم اطلاعات از یک سو و پیشرفت فن‌آوری‌های نوین از سوی دیگر سبب شده تا طیف وسیعی از تهدیدها و جرم‌ها در پایگاه‌داد مطرح گردد. وجود این تهدیدها و جرم‌ها سبب شده تا تمهیدات امنیتی چون حفظ محرمانگی، صحت و دسترس‌پذیری در این سامانه‌ها از جایگاه ویژه‌ای برخوارد باشد. مکانیزم‌های امنیتی موجود به سبب حفظ کارایی سامانه و دلایلی از این دست نمی‌توانند جلوی تمامی تهدیدها و جرایم اینترنتی در پایگاه‌های داده را بگیرند و لذا امکان وقوع جرم در سامانه امکان‌پذیر است. از این‌رو، به منظور شناسایی شواهد جرم، جمع‌آوری اطلاعات مربوطه، تجزیه‌وتحلیل آن‌ها و در نهایت تهیه مستندات لازم جهت اثبات وجود جرم، شاخه‌ی جرم‌شناسی پایگاه‌داده مطرح می‌گردد. وجود زیرساخت‌های مختلف برای سامانه‌های پایگاه‌داده، طبیعت چند بعدی سامانه‌ها، ابزارهای مختلف جرم‌شناسی و فقدان مدیریت دانش مرتبط با جرم‌شناسی را می‌توان از جمله چالش‌های اصلی در این حوزه دانست. وجود این چالش‌ها سبب شده است تا جرم‌شناسی پایگاه‌داده به عنوان یک موضوع گسترده و پیچیده معرفی گردد.

در این راستا بر آن شدیم تا جرم‌شناسی در پایگاه‌داده MsSQL را برای برخی از عملکردهای پر کاربرد، مورد بحث و بررسی قرار دهیم. بدین منظور، با استفاده از فرآیند استاندارد جرم‌شناسی، مراحل شناسایی، گردآوری اطلاعات، تحلیل، ترمیم و ارائه مستندات کافی برای اثبات جرم را تشریح می‌کنیم. لازم به ذکر است که اگر چه جرم‌شناسی از حدود سال ۲۰۰۴ مطرح است، اما تا کنون پیشرفت خاصی در زمینه‌ی ابزارهای تجاری و رایگان برای رسیدگی به رویدادهای غیرمجاز ارائه نشده است.

در ادامه و در فصل اول، مفاهیم و تعاریف اولیه‌ی جرم‌شناسی پایگاه‌داده، چالش‌ها، اهداف و گام‌های اجرایی جرم‌شناسی تشریح می‌شود. در فصل دوم، فرآیند جرم‌شناسی و مدیریت جرم در هر سامانه پایگاه‌داده (سمپاد) مورد بحث و بررسی قرار می‌گیرد. در فصل‌های سوم تا ششم، گام‌های شناسایی، جمع‌آوری شواهد، استخراج و تحلیل اطلاعات، ترمیم و ارائه مستندات مربوط به جرم، به ترتیب برای هر یک از رویدادهای درج، حذف، مشاهده غیرمجاز محتوای جداول، ویرایش غیرمجاز محتوای جداول ، تغییر غیرمجاز شمای پایگاه‌داده و تلاش برای ورود غیرمجاز به سامانه پایگاه‌داده مطالعه می‌شود. لازم به ذکر است که تمرکز ما در فرآیند جرم‌شناسی پایگاه‌داده تنها بر روی اطلاعات حاصل از رویدادنگاری و ممیزی در پایگاه‌داده است و شامل اطلاعات موجود در سیستم عامل و حافظه‌ی اصلی نمی‌باشد و همچنین کلیه پیکربندی‌های ارائه شده در مستند حاضر بر روی MsSQL server 2012 نسخه‌ی Express و نسخه Evaluation می‌باشد. لازم به ذکر است که علت بهره‌گیری از دو نسخه متفاوت، پوشش دادن سایر منابعی است که به طور بالقوه توسط پایگاه‌داده MsSQL فراهم شده و می‌تواند در فرآیند جرم‌شناسی استفاده شود.

فهرست مطالب

۱ جرم شناسی پایگاه‌داده

بسیاری از سازمان‌ها و آژانس‌های دولتی از پایگاه‌داده برای ذخیره و بازیابی اطلاعات استفاده می‌کنند. از آنجاییکه داده‌های سازمان‌ها می‌توانند حاوی اطلاعات حساس و حیاتی باشند، لذا حفاظت از پایگاه‌داده به عنوان یک سامانه ذخیره‌سازی اطلاعات، یک امر حیاتی و مهم به حساب می‌آید. داده‌های ذخیره شده در پایگاه‌داده ممکن است توسط کاربران غیرمجاز (از درون سازمان یا بیرون سازمان) تغییر داده شوند. لازم به ذکر است که بسیاری از مجرمان به دلیل فقدان دلایل کافی برای اثبات جرم، محکوم نمی‌شوند. در این شرایط، جرم‌شناسی نقش مهمی در ارایه روش‌های اثبات شده علمی برای جمع‌آوری اطلاعات، تحلیل و بررسی و نهایتا ارائه‌ی شرح مفصلی از فعالیت‌های مجرمانه‌ی سایبری ایفا می‌کند. جرم‌شناسی پایگاه‌داده، شاخه‌ای از جرم‌شناسی دیجیتال است که در آن پایگاه‌داده و فراداده‌های مرتبط با آن به صورت قانونی مورد مطالعه قرار می‌گیرند. یکی از مهم‌ترین اهدفی که در جرم‌شناسی پایگاه‌داده دنبال می‌شود آن است که تشخیص دهیم چه کسی، چه زمانی، چه داده‌ای را تغییر داده است. لازم به ذکر است که پایگاه‌داده قربانی معمولاً شامل اطلاعاتی است که در طول تحقیقات قانونی به کار می‌آید. در ادامه برخی از تعاریف و مفاهیم کلیدی، چالش‌ها، اهداف و گام‌های اجرایی فرآیند جرم‌شناسی در سامانه پایگاه‌داده، مورد بحث و بررسی قرار می‌گیرد.

۱-۱ تعاریف و مفاهیم

در این بخش، برخی از مهم‌ترین تعاریف و مفاهیم جرم‌شناسی پایگاه‌داده که در سرتاسر مستند مورد استفاده قرار گرفته است، مورد بحث و بررسی قرار می‌گیرد.

سامانه مدیریت پایگاه‌داده: به نرم افزاری اطلاق می‌شود که برای نگهداری و مدیریت حجم وسیعی از اطلاعات طراحی شده و مورد استفاده قرار می‌گیرد [۴].

جرم‌شناسی: مجموعه‌ای از آزمایش‌ها یا روش‌های علمی است که در تحقیقات جنایی مورد استفاده قرار می‌گیرد. امروزه جرم‌شناسی به روشی برای به دست آوردن شواهد جنایی به منظور ارائه در دادگاه اشاره دارد.

تجزیه و تحلیل جرم‌شناسی: به فرآیند بررسی رویدادهای غیرمجاز با در نظر گرفتن خط زمانی[۱]موجود از شواهد فیزیکی مربوط به آن، تجزیه و تحلیل جرم‌شناسی اطلاق می‌شود. نتایج حاصل می‌تواند در شناسایی مجرم کمک کند و همچنین به منظور ارایه شواهد برای اثبات جرم، مورد استفاده قرار گیرد [۱].

جرم‌شناسی دیجیتال: به فرآیندی که در آن از روش‌های علمی مرسوم و اثبات شده برای: (۱): حفاظت، (۲): جمع‌آوری، (۳): اعتبارسنجی، (۴): شناسایی، (۵): تجزیه‌وتحلیل، (۶): تفسیر، (۷): مستندسازی و ارائه‌ی شواهد دیجیتال به منظور تسهیل در بازسازی رویدادهای شناخته شده جنایی یا پیش‌بینی اقدامات غیرمجاز استفاده می‌شود، جرم شناسی دیجیتال اطلاق می‌گردد [۵].

جرم‌شناسی پایگاه‌داده: جرم‌شناسی پایگاه‌داده فرآیندی است که تلاش می‌کند تا زمان/چگونگی/ چرایی و عامل (های) رویداد غیرمجاز در سامانه را مشخص نماید. لازم به ذکر است که محتوای پایگاه‌داده، فراداده‌ها[۲] (به ویژه فایل‌های رویدادنگاری) و داده‌های موجود در حافظه از جمله مهمترین مولفه‌های تاثیرگذار در این فرآیند به حساب می‌آیند.

ممیزی:[۳]به نظارت و ثبت فعالیت‌های کاربران در پایگاه‌داده، ممیزی اطلاق می‌شود [۶].

رویدادنگاری:[۴] به تاریخچه‌ای از فعالیت‌های اجرا شده توسط سامانه مدیریت پایگاه‌داده اطلاق می‌شود که برای تضمین ویژگی‌های جامعیت (ACID[۵]) به هنگام خرابی سخت افزاری[۶]یا از کار افتادگی ناگهانی[۷]مورد استفاده قرار می‌گیرد [۷].

مصنوعات[۸]پایگاه‌داده: رکوردها یا اطلاعاتی هستند که از پایگاه‌داده قابل استخراج بوده و در تجزیه و تحلیل جرم‌شناسی مفید هستند [۲].

---

۱ Timeline

۲ Metadata

۳ Auditing

۴ Logging

۵ Atomicity-Consistency-Isolation-Durability

۶ Hardware failure

۷ Crash

رویداد غیرمجاز: به رویدادی اطلاق می‌شود که به صورت خصمانه یا ناخواسته در روال عادی سیستم تغییر نامطلوبی را ایجاد می‌کند.

۲-۱ چالش‌ها

جرم‌شناسی پایگاه‌داده، چالش‌های زیادی به همراه دارد که آن را تبدیل به یک موضوع پیچیده می‌کند. سامانه‌های پایگاه‌داده، سرویس‌ها و زیرساخت‌های مختلفی دارند که از یک پایگاه‌داده به پایگاه‌داده دیگر، متفاوت هستند. علاوه بر این، پایگاه‌های داده مختلف دارای مصنوعات جرم‌شناسی متفاوت همچون روش‌ها، مدل‌ها، چارچوب‌ها، ابزارها، فعالیت‌ها و خط‌مشی‌های مختلف هستند از سوی دیگر، سامانه‌های پایگاه‌داده دارای طبیعت چند بعدی شامل سطح داخلی، سطح مفهومی و سطح خارجی هستند. سطح داخلی شامل فایل فیزیکی است و سطح مفهومی، سطح منطقی است که زیرساخت منطقی شمای پایگاه‌داده ازجمله کاربران، جداول، شاخص‌ها و رویه‌ها را نمایش می‌دهد. سطح خارجی با کاربران واقعی سروکار دارد تا بتوانند داده‌ها را تغییر دهند. بنابراین، ابعاد مختلف پایگاه‌داده در جرم‌شناسی پایگاه‌داده ایفای نقش می‌کنند [۸].

یک چالش مهم دیگر در حوزه جرم‌شناسی پایگاه‌داده، تشخیص وجود نقض[۹] امنیتی در سامانه است. در واقع، فرآیند شناسایی و ترمیم به هنگام وقوع جرم، تا زمانی که شخصی فکر کند که نقض امنیتی رخ داده است، به تاخیر می‌افتد. در حالت کلی، شواهد برای وجود نقض امنیتی را می‌توان در سه دسته‌ی زیر خلاصه کرد:

- داده‌های سازمان در خارج از سازمان پیدا می‌شوند که مسلماً عادی و مجاز نیست.
- رویدادی مشاهده می‌شود که غیرمنتظره است؛ مثلاً فرآیندی در زمان اشتباه اجرا شده است یا دسترسی به سامانه، خارج از ساعت‌های اداری و مجاز اتفاق افتاده است.
- نشانه‌هایی از تغییر غیرمجاز داده‌ها وجود داشته باشد.

روش‌هایی برای جمع‌آوری و تجمیع شواهد مجرمانه در پایگاه‌داده وجود دارد. جرم‌شناسی پایگاه‌داده زمانی رخ می‌دهد که از مأمور ممیزی، نحوه وقوع نقض امنیتی و شخص مجرم، درخواست شود. روش‌هایی که برای جرم‌شناسی پایگاه‌داده وجود دارند، اصولاً دارای دو محدودیت زیر هستند:

---

[۸] Artifact

[۹] Breach

- کاربر پس از هفته‌ها یا ماه‌ها متوجه نقض امنیتی در پایگاه‌داده می‌شود. در این صورت داده‌های ناپایدار در پایگاه‌داده به عنوان شواهد وجود ندارد.
- ممکن است هیچ ممیزی برای پایگاه‌داده فعال نباشد.

دو عامل فوق‌الذکر سبب می‌شوند که بررسی رویداد مورد تقاضا، زمان آن و نتیجه‌ی حاصل از بررسی در پیچیده‌ترین حالت ممکن قرار گیرد. آنچه جرم‌شناسی پایگاه‌داده را از جرم‌شناسی شواهد فیزیکی متمایز می‌کند، حجم پایگاه‌داده و نیاز به در حال اجرا ماندن آن در محیط عملیاتی است.

۱-۳ اهداف

برخی از مهم‌ترین اهدافی که در جرم‌شناسی پایگاه‌داده به دنبال آن هستیم به قرار زیر است [۸-۹]:

ردگیری[۱] اعمال DML و DDL: در چرخه حیات پایگاه‌های داده بارها نیاز می‌شود که تغییرات در داده‌ها همچون درج، بروزرسانی و حذف داده‌ها که از اعمال دستکاری داده (DML) به حساب می‌آیند و تغییرات در اشیای پایگاه‌داده همچون ایجاد، تغییر و حذف یک جدول که از اعمال تعریف داده (DDL) به حساب می‌آیند، ردگیری و بررسی شوند. با این کار، رویدادهای غیرمجاز تشخیص داده شده و مجرمان شناسایی می‌شوند.

شناسایی داده‌ها پیش و پس از تراکنش: در طول یک تراکنش، ممکن است داده‌ها دستخوش تغییرات زیادی شوند. گاهاً داده‌های جدیدی ایجاد، داده‌های موجود حذف یا تغییر داده می‌شوند. شناسایی تغییرات اعمال شده بر روی داده‌ها، ما را در تشخیص رویدادهای غیرمجاز کمک خواهد کرد.

بازگشت به عقب اعمال غیرمجاز تغییر داده: در صورتی که داده‌ها طی رویدادهای غیرمجاز تغییر داده شوند، باید بتوان آن‌ها را به وضعیت پیش از رویداد غیرمجاز بازگرداند. همچنین در صورت حذف داده‌ها، نیاز به بازیابی داده‌های حذف شده خواهد بود.

اثبات یا رد وقوع نقض امنیتی: یک چالش مهم در حوزه جرم‌شناسی پایگاه‌داده، تشخیص وجود نقض امنیتی در سامانه است. در واقع فرآیند شناسایی و ترمیم به هنگام وقوع جرم، تا زمان تشخیص نقض امنیتی

---

۱ Trace

در سامانه به تاخیر می‌افتد. پس از آن با طی کردن گام‌های مطرح در فرآیند جرم‌شناسی پایگاه‌داده می‌توان ثابت کرد که نقض امنیتی رخ داده است یا خیر.

تعیین محدوده‌ی نفوذ به پایگاه‌داده: هنگامی که به پایگاه‌داده حمله می‌شود، تشخیص حمله‌ی انجام گرفته و محدوده‌ی نفوذ به منظور شناسایی خسارات وارد شده و محدوده‌ی تغییرات غیرمجاز، از اهمیت بالایی برخودار است.

کشف اینکه چه اتفاقی در چه زمانی رخ داده است: با بررسی و تحلیل رویدادهای ثبت شده، اطلاعاتی همچون کاربر اجراکننده رویداد، زمان رخداد، داده متاثر از رویداد، چگونگی تغییر و علت آن مشخص می‌شود. با دانستن این اطلاعات، نه تنها جزییات رویدادهای رخ داده مشخص می‌شوند بلکه می‌توان با توجه به توالی زمانی رویدادها و تجمیع آن‌ها، اطلاعات جدیدی نیز کسب کرد. برخی از داده‌ها به تنهایی دارای ارزش کمی هستند ولی هنگامی که با سایر اطلاعات ترکیب می‌شوند، می‌توانند نقض امنیتی را آشکارتر سازند.

۱-۴ گام‌های اجرایی

گام‌هایی که برای جرم‌شناسی پایگاه‌داده باید برداشته شوند به موقعیت و نوع سامانه مدیریت پایگاه‌داده‌ی وابسته است. در ادامه، برخی از مهم‌ترین گام‌هایی که در این راستا می‌بایست لحاظ شود، آورده شده است.

۱. **مرحله‌ی شناسایی:** در مرحله‌ی شناسایی، رویداد و نوع آن با توجه به نشانه‌های موجود در سامانه شناسایی می‌شود. این مرحله از آن جهت مهم است که سایر مراحل را تحت تأثیر قرار می‌دهد. برخی از مهمترین اهدف این مرحله، شناسایی مفاهیم مرتبط با بررسی جرم‌شناسی همچون منابع پایگاه‌داده، منابع سیستم‌عامل، منابع شبکه، تیم‌های بررسی، روش‌های بررسی، محیط بررسی، خط‌مشی‌ها، قوانین و مجوزها هستند.

۲. **تعیین روش جمع‌آوری داده‌های جرم‌شناسی:** به منظور بررسی پایگاه‌داده آسیب‌دیده یا مورد سوءاستفاده، سه روش جمع‌آوری داده به شرح زیر وجود دارد:

- **جمع‌آوری داده‌ها به صورت زنده**[۱]: این روش جمع‌آوری داده زمانی اتفاق می‌افتد که سامانه مورد تجزیه و تحلیل به طور همزمان در حال سرویس‌دهی نیز می‌باشد.
- **جمع‌آوری داده‌ها به صورت غیرزنده**[۲]: روش جمع‌آوری داده به صورت غیرزنده، شامل نسخه‌برداری داده‌ها از سیستم مورد بررسی است.
- **جمع‌آوری داده‌ها به صورت ترکیبی**[۳]: روش جمع‌آوری داده‌ها به صورت ترکیبی با بهره‌گیری از ویژگی‌های کلیدی هر دو روش قبل، ترکیبی از این دو روش را برای جمع‌آوری داده در پیش می‌گیرد.

لازم به ذکر است که صرف‌نظر از روش مورد استفاده می‌بایست این اطمینان حاصل شود که شواهد دیجیتالی حفظ و نگهداری می‌شوند و داده‌ها به صورت ناخواسته تغییر نمی‌کنند یا از بین نمی‌روند.

۳. **جمع‌آوری مصنوعات ناپایدار[۴] و پایدار:** مصنوعات و اطلاعات مختلف را می‌توان از پایگاه‌داده، سیستم‌عامل، سرورهای وب یا فایل‌های رویدادنگاری استخراج کرد. لازم به ذکر است که جمع‌آوری مصنوعات و شواهد در سامانه می‌تواند سبب تغییر در پایگاه‌داده شود. از این‌رو، پیش از استخراج اطلاعات از پایگاه‌داده باید نسبت به این موضوع و پایدار یا ناپایدار بودن اطلاعات، آگاهی پیدا کرد. هر پایگاه‌داده شواهد مربوط به اعمال مختلف را در فایل‌های رویدادنگاری مختلفی پایگاه‌داده ذخیره می‌کند. این بدین معناست که برای تجزیه و تحلیل جرم‌شناسی می‌بایست نسبت به چگونگی عملکرد پایگاه‌داده، محل فایل‌ها و مصنوعات مختلف اطلاع داشت. مصنوعات در سطح پایگاه‌داده به دو دسته تقسیم می‌شوند: ۱): داده‌های فرار و ۲): داده‌های غیر فرار که در ادامه به تشریح هر یک از آنها می‌پردازیم.

- **داده‌های فرار:** به برخی از داده‌ها که به منظور افزایش سرعت، قابلیت اطمینان و بهره‌وری پایگاه‌داده درحافظه‌های ناپایدار ذخیره می‌شوند، داده‌های فرار اطلاق می‌شود. به عنوان

---

[۱] Live acquisition

[۱] Dead acquisition

[۱] Hybrid acquisition

[۱] Volatile artifacts

نمونه، پایگاه‌داده اوراکل اطلاعات زیادی را در SGA[۱۵] به منظور افزایش کارایی ذخیره می‌کند. لازم به ذکر است که به طور معمول این دسته از داده‌ها دائما و با سرعت بالایی در حال تغییر هستند.

- **داده‌های غیرفرار:** به رکوردهای ذخیره شده در پایگاه‌داده، داده‌های غیرفرار یا پایدار اطلاق می‌شود.

ذکر این نکته ضروری است که فرآیند جرم‌شناسی در سامانه پایگاه‌داده نباید تنها بر روی پایگاه‌داده متمرکز باشد. پایگاه‌داده در یک محیط مجزا در حال اجرا و سرویس‌دهی نیست و متکی به زیرساخت مهمی چون سیستم عامل است. بنابراین باید سایر مصنوعات و اطلاعات جانبی از سیستم عامل‌ها و رویدادهای ثبت شده در سرور را نیز جمع‌آوری و آنها را بررسی نمود.

۴. **حفاظت[۱۶] و احراز اصالت[۱۷] داده‌های جمع‌آوری شده:** هدف از این مرحله آن است که مقدار شواهدی که مجدداً بر روی آن‌ها اطلاعاتی نوشته می‌شود، کاهش پیدا کند. مراقبت‌های شدیدی برای تضمین عدم تغییر غیرمنتظره داده‌ها باید انجام شود. اگر چه داده‌ها را می‌توان با ارسال پرسمان به پایگاه‌داده‌ی تغییریافته به دست آورد، باید از اجرای هر نوع پرسمانی که باعث حذف اطلاعات از پایگاه‌داده شود، اجتناب نمود. علاوه بر این، در صورت وجود پایگاه‌داده آسیب‌دیده یا مورد سوء استفاده قرار گرفته، صرف‌نظر از روش بدست آوردن داده، هیچ عبارت SQL ای نباید اجرا شود؛ زیرا باعث تغییر داده‌های ذخیره‌شده در حافظه و صفحات داده مربوط به پایگاه‌داده می‌شود. این کار همچنین باعث تقسیم شدن صفحه داده داخلی و محل ذخیره‌سازی داده‌های جدید در حافظه پنهان[۱۸] می‌شود و فرآیند بررسی را پیچیده‌تر می‌کند. بنابراین باید پیش از فرآیند جمع‌آوری، نسخه پشتیبان از داده‌ها و فایل‌های مهم تهیه گردد. لازم به ذکر است که روش‌ها و ابزارهای مورد استفاده برای جمع‌آوری اطلاعات و شواهد می‌بایست تا حد امکان قابل اطمینان باشند.

---

[۱۵] قسمتی از حافظه‌ی سیستم که میان تمامی پردازه‌های مربوط به یک نمونه‌ی واحد از پایگاه‌داده‌ی اوراکل مشترک است.

[۱۶] Preservation

[۱۷] Authentication

[۱۸] Cache

۵. **تجزیه و تحلیل شواهد و تعیین فعالیت‌های مهاجم:** تجزیه و تحلیل داده‌های جمع‌آوری شده به نوع داده‌ها، سامانه پایگاه‌داده و رویداد خاصی که قرار است مورد بررسی قرار گیرد، بستگی دارد. مرحله‌ی تجزیه و تحلیل می‌بایست ابعاد مربوط به هر رویداد و محلی که اطلاعات مربوطه یافت می‌شود را در نظر بگیرد. بنابراین در مرحله‌ی تجزیه و تحلیل، اطلاعاتی همچون شخص مجرم، زمان ارتکاب جرم، داده هدف، دلایل ارتکاب و نحوه اجرای جرم تعیین می‌گردد. تجمیع داده‌ها در فرآیند جرم‌شناسی پایگاه‌داده از اهمیت بسزایی برخوردار است. برخی از داده‌ها به تنهایی دارای ارزش کمی هستند اما هنگامی که با سایر اطلاعات ترکیب می‌شوند، می‌توانند نقض امنیتی را آشکارتر سازند.

۶. **بازسازی پایگاه‌داده:** در بررسی پایگاه‌داده آسیب‌دیده یا مورد سوءاستفاده، داده‌هایی که قبلاً حذف شده‌اند، باید بازیابی و اقدامات انجام شده توسط مهاجم شناسایی شوند.

۷. **ارائه مستندات:** در مرحله آخر، کلیه‌ی بررسی‌های صورت گرفته در یک قالب استاندارد مستند و به مدیر سامانه و دادگاه ارائه می‌شود. لازم به ذکر است که مستندات تهیه شده برای سایر بررسی‌کنندگان که سناریوی مشابه‌ای را تجربه می‌کنند و همچنین برای حفاظت از پیگردهای قانونی بررسی‌کنندگان در آینده مفید خواهد بود.

۱-۵ جمع‌بندی

در این فصل به طور مشروح به معرفی مفاهیم جرم‌شناسی پایگاه‌داده و همچنین بررسی چالش‌ها، اهداف و گام‌های اجرایی در فرآیند جرم‌شناسی پایگاه‌داده پرداخته شد. در این راستا و در ابتدا، وجود تنوع‌های گسترده از سامانه‌ها، سطوح مختلف داخلی، مفهومی و خارجی پایگاه‌داده و نحوه تشخیص وقوع جرم به عنوان مهم‌ترین چالش مطرح در حوزه جرم‌شناسی پایگاه‌داده بررسی گردید. در ادامه، برخی از مهم‌ترین اهداف جرم‌شناسی پایگاه‌داده، تحت عناوینی چون شناسایی و اثبات وقوع جرم، کشف رویداد غیرمجاز و زمان وقوع آن، تعیین محدوده نفوذ و بازگرداندن وضعیت سامانه به وضعیت قبل از وقوع جرم معرفی گردید. در نهایت نیز، گام‌های اجرایی فرآیند جرم‌شناسی پایگاه‌داده از مرحله شناسایی جرم تا ارائه مستندات مربوط به شواهد وقوع جرم، تشریح گردید.

۲ شناسایی و مدیریت جرم

پژوهش‌های انجام شده در زمینه‌ی جرم‌شناسی دیجیتال منجر به توسعه‌ی روش‌ها و مدل‌های فرآیندی مختلفی شده است. بسیاری از این روش‌ها به سبب ویژگی‌های خاص هر یک از سامانه‌های پایگاه‌داده، به طور کامل قابل انطباق با جرم‌شناسی پایگاه‌داده نبوده و می‌بایست در آن‌ها تغییراتی صورت پذیرد. با استفاده از مدل‌های فرآیند جرم‌شناسی پایگاه‌داده می‌توان به صورت ساختاریافته، عملیات مربوط به جرم‌شناسی پایگاه‌داده را پیش برد. در این فصل، ابتدا تمهیدات مورد نیاز برای فراهم کردن بستر مناسب برای

جرم‌شناسی در پایگاه‌داده مورد مطالعه قرار می‌گیرد. در ادامه و در بخش دوم نیز به شرح و بررسی فرآیند جرم‌شناسی در سامانه پایگاه‌داده می‌پردازیم. در بخش پایانی نیز ملاحظات و رهنمون‌های مورد نیاز برای رهگیری و انجام مراحل موجود در فرآیند جرم‌شناسی تشریح می‌گردد.

۲-۱ تمهیدات جرم‌شناسی

در این بخش، تمهیدات لازم برای جرم‌شناسی پایگاه‌داده را در طیف وسیعی از نیازمندی‌ها از پیش از وقوع جرم تا نیازمندی‌های نهایی مربوط به ارائه مستندات و اثبات وقوع جرم، مورد بحث و بررسی قرار می‌دهیم. لازم به ذکر است که داشتن طرح و برنامه دقیق، مهمترین گام در فرآیند جرم‌شناسی است. در یک دسته‌بندی کلی می‌توان تمهیدات مورد نیاز برای فرآیند جرم‌شناسی پایگاه‌داده را در موارد زیر خلاصه کرد [۱۵-۱۷]:

۱. **تعیین مدیر فرآیند:** این تمهید پیش از وقوع جرم انجام می‌شود. پیش از آنکه جرمی رخ دهد، شخصی باید به عنوان هماهنگ‌کننده تعیین شود. این شخص باید فرآیند تجزیه و تحلیل جرم‌شناسی را رهبری کند و از مستند تهیه شده از فرآیند به عنوان یک چک لیست[۹] برای اطمینان از اجرای گام به گام فرآیند استفاده نماید. علاوه بر این می‌بایست تیمی تحت رهبری هماهنگ‌کننده نیز وجود داشته باشد که از مهارت‌های امنیتی و مدیریتی در حوزه پایگاه‌داده برخوردار باشند. همچنین وجود یک مجموعه از ابزارها برای استفاده در فرآیند تجزیه و تحلیل بسیار مهم است. هریک از ابزارها و هر آنچه که هر ابزار انجام می‌دهد، می‌بایست مستند شود. بزرگترین چالش در فرآیند تجزیه و تحلیل، عدم وجود ابزارهای استاندارد رایگان به منظور تجزیه و تحلیل فایل‌های ردیابی[۱۰] و رویدادهای تولید شده توسط پایگاه‌داده است. در صورتی که در طول فرآیند جرم‌شناسی از ابزاری استفاده شود، باید بتوان ثابت کرد که ابزارهای مورد استفاده، شواهد جمع‌آوری شده را تغییر نمی‌دهند و حذف نمی‌کنند. در صورتی که ابزار توسط خود افراد ایجاد شده باشد، اثبات عدم تغییر در شواهد، می‌تواند دشوار باشد. بنابراین بهتر است از ابزارهای تجاری از پیش تعیین شده‌ای که در دادگاه‌ها قابل قبول و استناد هستند، استفاده شود. لازم به ذکر است که

---

۹ Checklist

۱۰ Trace files

ابزارهای مورد استفاده نیز می‌بایست از حیث امنیتی قابل اعتماد باشند و اجازه ندهند مهاجم از طریق آن‌ها به شواهد موجود خدشه‌ای وارد کند.

۲. **تعیین مولفه مرکزی برای اطلاع‌رسانی:** تعیین مولفه‌ی مرکزی برای گزارش جرم‌های حادث شده به عنوان یک تمهید پیش از وقوع جرم، از اهمیت بالایی برخوردار است. وجود این مولفه سبب می‌شود تا ذینفعان به سرعت از وقوع نقض امنیتی مطلع و بررسی آن را در دستور کار خود قرار دهند.

۳. **تشخیص وقوع نقض امنیتی:** به محض تشخیص نقض امنیتی در سامانه می‌بایست اطلاعات کافی برای مولفه مرکزی ارسال گردد. مولفه مرکزی نیز اطلاع‌رسانی‌های لازم را در این رابطه برای سایر ذینفعان ارسال می‌کند.

۴. **انتقال کنترل فرآیند به مدیر هماهنگ‌کننده:** به محض تشخیص وقوع نقض امنیتی در سامانه و اعلام آن توسط مولفه مرکزی، کنترل فرآیند به مدیر تجزیه و تحلیل جرم‌شناسی منتقل می‌شود تا این اطمینان حاصل شود که فرآیند به درستی و با دقت توسط تیم دنبال می‌شود.

۵. **پرهیز از قطع اتصال شبکه و خاموش کردن سامانه:** در این گام، به هیچ وجه نباید سامانه پایگاه‌داده خاموش یا اتصال آن به شبکه قطع گردد. توجه به این نکته حائز اهمیت است که داده‌ها و شواهد ناپایدار با خاموش شدن سیستم از بین می‌روند. اینکه مهاجم از ادامه‌ی فعالیت‌های مخرب باز بماند، ایده‌ی خوبی است ولی از دست دادن شواهد ناپایدار، بررسی‌های آتی را ممکن است با مشکل روبرو کند.

۶. **بررسی واقعی بودن حمله:** در این مرحله می‌بایست وجود نقض امنیتی و حمله به سامانه بررسی گردد. لازم به ذکر است که بررسی در دسترس بودن داده‌های حساس به طور ویژه برای مهاجم و در حالت کلی در بستر عمومی وب از اهمیت بالایی در این مرحله برخوردار است.

۷. **جمع‌آوری داده‌های فرار:** در این گام، رسیدگی به نقض امنیتی را آغاز نموده و داده‌های فرار را از روی سرور پایگاه‌داده جمع‌آوری می‌نماییم. از جمله داده‌های فرار می‌توان به اطلاعات مربوط به کاربران وارد شده به سامانه، پردازه‌های در حال اجرا، پورت‌های و فایل‌های باز، اشاره کرد. همچنین تمامی فایل‌های رویدادنگاری پایگاه‌داده، فایل‌های ردیابی و فایل‌های پیکربندی نیز باید جمع‌آوری و از آن‌ها به درستی نسخه‌برداری شود. علاوه بر این، از رویدادهای ثبت شده توسط سرور وب و برنامه‌ی کاربردی و سایر فایل‌های رویدادنگاری مهم نیز باید نسخه‌ای تهیه شود. اطلاعات موجود در حافظه نیز باید تخلیه و ثبت شوند. به عنوان نمونه، در پایگاه‌داده اوراکل اطلاعات موجود در SGA باید تخلیه و ثبت شوند. می‌توان آخرین پرسمان SQL اجرا شده را از SGA به دست آورد. در صورتی که بررسی جرم‌شناسی خیلی سریع انجام شود، شانس این وجود دارد که عبارت‌های SQL

که قسمتی از حمله بوده‌اند، در SGA وجود داشته باشند. همچنین می‌توان نشست‌ها و پردازه‌های موجود در SGA را نیز استخراج و جمع‌آوری کرد. تقریباً هرکاری که در پایگاه‌داده انجام می‌شود، آن را تغییر می‌دهد. بنابراین استخراج شواهد از پایگاه‌داده آسیب‌دیده می‌بایست با دقت و با ترتیب درستی انجام شود. لازم به ذکر است که داده‌هایی که بیشتر در معرض تغییر قرار دارند، باید سریعتر استخراج شوند.

۸. **جمع‌آوری داده‌های غیرفرار:** پس از جمع‌آوری داده‌های فرار که حساسیت بیشتری برای جمع‌آوری اطلاعات نسبت به سایر اطلاعات را دارند، سایر شواهد را نیز از پایگاه‌داده جمع‌آوری می‌کنیم. از جمله این شواهد می‌توان به لیست کاربران، مجوزهای کاربران و عضویت در نقش‌ها اشاره کرد.

۹. **قطع اتصال پایگاه‌داده به شبکه:** پس از جمع‌آوری اطلاعات و شواهد مورد نیاز برای بررسی جرم، به منظور کاهش مخاطرات احتمالی ناشی از آن می‌بایست اتصال سامانه به شبکه را قطع نمود.

۱۰. **تهیه نسخه پشتیبان:** در صورت امکان از کل دیسک سخت افزاری و یا در صورت وجود محدودیت، از شواهد موجود در سرور پایگاه‌داده، نسخه پشتیبان تهیه شود.

۱۱. **انجام تجزیه و تحلیل بر روی داده‌ها:** در این گام، بر روی داده‌ها و شواهد جمع‌آوری شده تجزیه و تحلیل صورت می‌گیرد و سعی می‌شود تا حد امکان زمان شروع و خاتمه‌ی حمله به دست آید. لازم به ذکر است که زمان‌های به دست آمده ممکن است با بررسی‌های بیشتر، تغییر نمایند.

۱۲. **ایجاد جدول زمانی[۱] از رویدادها:** جدول زمانی، تمامی اطلاعات مربوط به اعمال انجام شده بر روی پایگاه‌داده را در خود جای داده است. بدین ترتیب می‌توان پی برد که:

- مهاجم چگونه به سیستم دسترسی پیدا کرده است،
- از طریق چه نام کاربری وارد شده است،
- چه اطلاعاتی را مشاهده نموده است،
- چه اعمالی را در پایگاه‌داده انجام داده است،
- با چه مجوزهایی به سیستم وارد شده است،
- و چه کارهای بیشتری را با مهارت بیشتر می‌توانست در سامانه اعمال کند.

---

۱ Timeline ۲

۱۳. **خاموش کردن سیستم و بازگرداندن[۲] آن به وضعیت پیش از حمله:** در این گام می‌بایست با توجه به میزان اهمیت پایگاه‌داده، برای خاموش کردن آن تصمیم‌گیری شود. پیش از خاموش کردن یا بازگرداندن پایگاه‌داده، باید کاملاً مشخص شود که مهاجم چه کارهایی را انجام داده است. در صورتی که داده‌ها تنها توسط مهاجم خوانده شده باشند و هیچ تغییری در آن‌ها اعمال نشده باشد، نیازی به بازگرداندن پایگاه‌داده به نقطه‌ای پیش از حمله نیست.

۱۴. **تهیه مستند از حمله:** مستندسازی فرآیند و کلیه‌ی شواهد جمع‌آوری شده، بسیار مهم است. همچنین باید یافته‌ها و آنچه با بررسی به دست آمده‌است نیز مستند شود. تمامی اعمال مهاجم به همراه نشانه‌هایی از سرقت داده‌ها باید ثبت شوند. ثبت اطلاعاتی همچون سیستم عامل سِرور و نسخه‌ی آن، نوع و نسخه‌ی پایگاه‌داده، رویداد غیرمجاز، نوع رویداد (خصمانه/ناخواسته)، شیوه ممیزی، منابع رویدادنگاری، شیوه یا ابزار تحلیل و امکان ترمیم در یک قالب استاندارد ضروری است. این مستندات برای شناسایی نقاط ضعف سامانه از اهمیت ویژه‌ای برخوردار است. بنابراین شناسایی نقاط ضعف سامانه و پیشنهادهایی برای حل آن‌ها نیز در این مستند ثبت می‌شوند.

۱۵. **گزارش رویدادها و فرآیند طی شده:** پس از تهیه‌ی مستند از رویدادها و فرآیند طی شده، مجموعه مستند گردآوری شده قابل ارائه به دادگاه یا سایر مقامات قانونی است.

۲-۲ فرآیند جرم‌شناسی

با استفاده از مدل‌های فرآیند جرم‌شناسی پایگاه‌داده می‌توان به صورت ساختاریافته‌ای عملیات مربوط به جرم‌شناسی پایگاه‌داده را پیش برد. در ابتدا و پیش از انجام هر عملی، باید نسبت به وقوع رویداد غیرمجاز اطمینان حاصل کرد و آن رویداد را شناسایی نمود. به عنوان مثال، در صورتی که مدیر پایگاه‌داده نسبت به حذف برخی از نام‌های کاربری از پایگاه‌داده مطلع شود، فرآیند جرم‌شناسی برای یافتن اطلاعاتی پیرامون این رویداد و ترمیم داده‌های حذف شده، آغاز می‌گردد.

پس از شناسایی وقوع رویداد غیرمجاز می‌بایست شواهد و اطلاعات پیرامون رویداد، جمع‌آوری گردد. با توجه به اینکه پایگاه‌داده در یک محیط مجزا نبوده و با سیستم عامل و برنامه‌های کاربردی در ارتباط است، می‌توان شواهد را از منابع مختلف به دست آورد. در جمع‌آوری اطلاعات و شواهد توجه به این نکته حائز اهمیت است

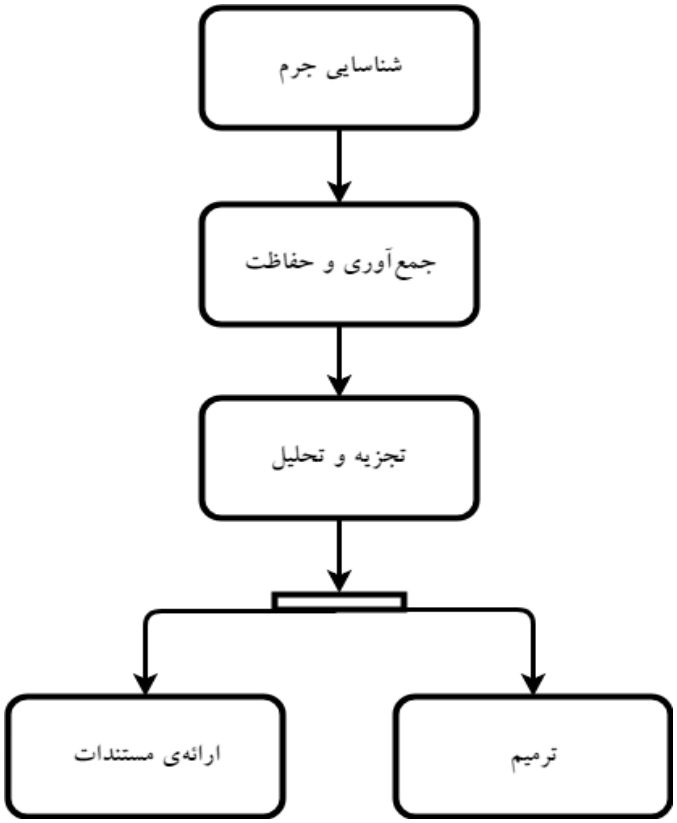
[۲] Restore [۲]

که برخی از اطلاعات همچون داده‌های موجود در حافظه، فرار بوده و هر لحظه احتمال از دست رفتن آن‌ها وجود دارد؛ بنابراین، اینگونه از اطلاعات باید هرچه سریعتر و پیش از فرا رسیدن موعد حذف، جمع‌آوری شوند. پس از آن، می‌توان اطلاعات پایدار و غیرفرار را استخراج و در مکانی امن نگهداری کرد.

اطلاعات و شواهد جمع‌آوری شده در صورتی برای تجزیه و تحلیل و همچنین ارائه در دادگاه‌های قانونی قابل قبول هستند که به درستی حفاظت شده باشند و بتوان ثابت کرد که در حین فرآیند جرم‌شناسی تغییری در آنها صورت نگرفته است.

در ادامه و در مرحله‌ی تجزیه‌وتحلیل، با بررسی و تحلیل داده‌های جمع‌آوری شده، اطلاعاتی همچون چه‌کسی تغییر را ایجاد کرده، چه‌زمانی تغییر رخ داده، چه داده‌ای تغییر داده شده، چرا و چگونه تغییر رخ داده است، مشخص می‌شود. لازم به ذکر است که تجمیع داده‌ها در فرآیند جرم‌شناسی پایگاه‌داده از اهمیت بالایی برخوردار است. برخی از داده‌ها به تنهایی دارای ارزش کمی هستند ولی هنگامی که با سایر اطلاعات ترکیب می‌شوند، می‌توانند نقض امنیتی را آشکار سازند. تمامی شواهد جمع‌آوری شده و فرآیند طی شده در جرم‌شناسی می‌بایست مستند شود. مستندات و گزارش‌های تهیه شده قابل ارائه به مدیریت سازمان و دادگاه در صورت نیاز خواهند بود.

به سبب گستردگی حوزه جرم‌شناسی پایگاه‌داده، تمرکز ما در فرآیند جرم‌شناسی پایگاه‌داده تنها بر روی اطلاعات ثبت‌شده در پایگاه‌های داده می‌باشد و استفاده از اطلاعات ثبت‌شده بر روی سیستم عامل، داده‌های موجود در حافظه و شبکه نادیده گرفته شده است. لازم به ذکر است که برای داشتن طرحی جامع در این زمینه، در نظر گرفتن کلیه مولفه‌های درگیر ضروری است. شکل ۱، گام‌های فرآیند جرم‌شناسی پایگاه‌داده در رویکرد اتخاذی را به تصویر کشیده است. در ادامه، هر یک از این گام‌ها مورد بحث و بررسی قرار گرفته است.



شکل ۱: فرآیند پیشنهادی جرم‌شناسی پایگاه‌داده منطبق با فرآیند استاندارد

شناسایی جرم:

فرآیند جرم‌شناسی با مشاهده رویدادهای غیرمجاز در سامانه آغاز می‌شود. منظور از رویدادهای غیرمجاز، رویدادهای خصمانه یا ناخواسته‌ای هستند که مدیر پایگاه‌داده انتظار وقوع آن‌ها را در شرایط فعلی ندارد. به عنوان مثال، در صورتی که مدیر پایگاه‌داده کاربرانی را در جدولی از پایگاه‌داده تعریف کرده باشد، درج کاربر جدیدی که توسط مدیر انجام نشده است، یک رویداد غیرمجاز به شمار می‌آید.

برخی از رویدادهای قابل بررسی در فرآیند جرم‌شناسی در سامانه مدیریت پایگاه‌داده عبارتند از:

- **درج، حذف و مشاهده‌ی غیرمجاز محتوای جداول:** رویدادهای غیرمجازی هستند که به صورت خصمانه یا ناخواسته توسط کاربر/برنامه‌کاربردی بر روی پایگاه‌داده اعمال می‌شوند و موجب درج سطر (های) جدید، حذف سطر (های) موجود یا مشاهده تمامی یا بخشی از جدول (ها) در

پایگاه‌داده می‌گردند. سه رویداد فوق، از جمله دستورات دستکاری داده (DML)[۲۴] در پایگاه‌داده به حساب می‌آیند.

- **بروزرسانی غیرمجاز داده‌های جداول**: رویداد غیرمجازی است که به صورت خصمانه یا ناخواسته توسط کاربر/برنامه‌کاربردی بر روی پایگاه‌داده اعمال می‌شود و سبب تغییر سطر (های) موجود می‌گردد. این رویداد نیز از جمله دستورات DML در پایگاه‌داده است. لازم به ذکر است که به سبب اهمیت بروزرسانی داده‌ها و چالش‌هایی که برای شناسایی وقوع جرم وجود دارد، این رویداد در برخی از سمپادها از دیگر رویدادهای مربوط به دستورات دستکاری داده متمایز شده است.
- **تغییر غیرمجاز شمای پایگاه‌داده:** رویداد غیرمجازی است که به صورت خصمانه یا ناخواسته توسط کاربر/برنامه‌کاربردی بر روی پایگاه‌داده اعمال می‌شود و سبب تغییر شمای جدول (ها) در پایگاه‌داده می‌گردد. این رویداد توسط دستورات تعریف داده (DDL)[۲۴] در پایگاه‌داده قابل اعمال است.
- **تلاش برای ورود غیرمجاز به پایگاه‌داده:** رویداد غیرمجازی است که به صورت خصمانه یا ناخواسته توسط کاربر/برنامه‌کاربردی بر روی پایگاه‌داده اعمال می‌شود و به طور معمول دو هدف زیر را دنبال می‌کند:
  - حدس نام کاربری و کلمه عبور کاربران مجاز جهت ورود به سامانه.
  - تلاش برای شناسایی شناسه یکتای سامانه[۲۵] (SID) و نهایتا دسترسی به حساب‌های کاربری و اعمال نفوذ در سامانه.

لازم به ذکر است که در برخی سمپادها، این تلاش تنها از طریق حدس نام‌کاربری و کلمه عبور کاربران مجاز جهت ورود به سمپاد امکان‌پذیر است.

---

۲ Data Manipulation Language

۲ Data Definition Language

۲ System Identifier

- **تغییر غیرمجاز در فایل‌های رویدادنگاری و ممیزی:** رویداد غیرمجازی است که به صورت خصمانه یا ناخواسته توسط کاربر/برنامه‌کاربردی بر روی پایگاه‌داده اعمال می‌شود و هدف آن از بین بردن فایل‌های رویدانگاری به منظور پاک کردن شواهد رویدادهای مجرمانه در پایگاه‌داده است.

جمع‌آوری اطلاعات و شواهد:

مصنوعات و اطلاعات مختلف برای شناسایی رویدادها را می‌توان از پایگاه‌داده، سیستم عامل، سرورهای وب یا فایل‌های رویدادنگاری استخراج کرد. جمع‌آوری مصنوعات و شواهد می‌تواند سبب تغییر در پایگاه‌داده شود. از این‌رو، پیش از استخراج اطلاعات از داخل یا خارج پایگاه‌داده می‌بایست نسبت به پایدار یا ناپایدار بودن اطلاعات آگاهی پیدا کرد. هر پایگاه‌داده شواهد مربوط به اعمال مختلف را در بخش‌ها و فایل‌های رویدادنگاری مختلف ذخیره می‌کند. این بدان معناست که به منظور تجزیه و تحلیل جرم‌شناسی می‌بایست نسبت به چگونگی عملکرد پایگاه‌داده، محل فایل‌ها و مصنوعات مختلف اطلاع داشت. از آنجا که تمرکز ما در فرآیند جرم‌شناسی پایگاه‌داده تنها بر روی اطلاعات حاصل از رویدادنگاری و ممیزی در پایگاه‌داده است، لذا اطلاعات مورد هدف برای گردآوری تنها به پایگاه‌داده محدود شده و شامل اطلاعات موجود در سیستم عامل و حافظه‌ی اصلی نمی‌باشد.

درصورتی که بر روی فایل‌های رویدادنگاری و ممیزی به صورت دوره‌ای اطلاعاتی ثبت شود، به هنگام شناسایی رویداد غیرمجاز می‌بایست ابتدا سرور پایگاه‌داده متوقف شود تا از اجرای پرسمان‌های جدید بر روی سرور خودداری گردد. بدین ترتیب از نگاشته شدن بر روی اطلاعات ثبت شده در فایل‌های رویدادنگاری و ممیزی جلوگیری می‌شود. در نهایت نیز به منظور انجام تحلیل‌های آتی می‌توان از فایل‌های مورد نیاز، یک نسخه‌ی پشتیبان تهیه نمود [۱۲].

مرحله جمع‌آوری اطلاعات و شواهد، شامل دو گام زیر می‌باشد:

۱. **تعیین نحوه‌ی جمع‌آوری اطلاعات:** در صورت مشاهده علائمی از وقوع رویداد غیرمجاز در پایگاه‌داده، با توجه به نوع رویداد می‌توان از منابع مختلفی برای جمع‌آوری اطلاعات استفاده کرد. به عنوان نمونه، برای جمع‌آوری اطلاعات مربوط به برخی از رویدادها می‌بایست از رویدادهای ثبت‌شده در فایل‌های رویدادنگاری و برای برخی دیگر می‌بایست از تعریف خط‌مشی‌هایی ممیزی استفاده کرد.
۲. **بررسی تنظیمات فعلی:** پس از آنکه منابع جمع‌آوری اطلاعات برای یک رویداد ناخواسته مشخص شدند، تنظیمات فعلی سیستم بررسی می‌شود. با بررسی اولیه‌ی تنظیمات مشخص می‌شود که آیا اقدامات اولیه برای ثبت اطلاعات قبل از وقوع رویداد انجام شده‌اند یا خیر. به عنوان نمونه، بررسی

می‌شود که آیا در پایگاه‌داده، خط‌مشی‌هایی برای ثبت اطلاعات ممیزی مربوط به جرم تعریف شده است یا خیر.

استخراج و تجزیه و تحلیل اطلاعات:

در این مرحله ابتدا ابزارها و پرسمان‌های هدف به منظور استخراج اطلاعات از منابع تعیین می‌شوند. یک روش محبوب در میان تحلیل‌گران جرم‌شناسی، استخراج عبارات SQL اجرا شده در پایگاه‌داده است. همچنین بی‌شک یکی از المان‌های بسیار مهم در تجزیه و تحلیل جرم‌شناسی، شناسایی هویت مجرم است. هر عملی که ثبت می‌شود را باید بتوان به یک شخص واقعی نسبت داد [۱۵]. سپس اطلاعات هدف از منابع مشخص شده استخراج و مرحله بررسی و تحلیل اطلاعات آغاز می‌گردد. در این مرحله با بررسی و تحلیل داده‌های جمع‌آوری شده، اطلاعاتی همچون عامل وقوع رویداد، زمان وقوع، دلایل وقوع، نوع تغییر حاصل از اجرای رویداد، داده متأثر از تغییر و چگونگی اعمال رویداد غیرمجاز، آشکار می‌شود [۱۰, ۱۱]. لازم به ذکر است که تجمیع داده‌ها در فرآیند جرم‌شناسی پایگاه‌داده بسیار حائز اهمیت است. به منظور سادگی در بررسی هر یک از رویدادهای ناخواسته، بخش استخراج اطلاعات و تجزیه و تحلیل با یکدیگر ادغام شده و تحت عنوان استخراج و تجزیه و تحلیل اطلاعات بیان می‌شود.

ترمیم:

پس از محرز شدن رخداد جرم در سامانه پایگاه‌داده، متناسب با نیاز و شواهد اطلاعاتی موجود، ممکن است ترمیم پایگاه‌داده امکان‌پذیر باشد. در واقع، ترمیم سامانه پایگاه‌داده بدین معناست که وضعیت پایگاه‌داده را به وضعیتی پیش از وقوع رویداد برگردانیم. به عنوان نمونه، در صورتی که داده‌های حساس از یک جدول حذف شده باشند، این مرحله به بازیابی داده‌های حذفی می‌پردازد.

ارائه‌ی مستندات:

در گام نهایی می‌بایست کلیه‌ی بررسی‌های صورت پذیرفته را در یک قالب استاندارد از پیش تعیین شده به نحوی مستند نمود که قابل ارائه به مدیریت سازمانی یا دادگاه قانونی برای طرح دعوی باشد. مستندسازی به سایر بررسی‌کنندگانی که سناریوی مشابه‌ای را تجربه می‌کنند نیز کمک خواهد کرد. به منظور ارائه مستندات مربوط به رویدادهای غیرمجاز کشف‌شده در سامانه، فرم استاندارد زیر ارایه شده است.

جدول ۱: تهیه‌ی مستند از فرآیند جرم‌شناسی پایگاه‌داده

عنوان رویداد			
وقوع جرم	عدم وقوع ☐	وقوع خصمانه ☐	وقوع ناخواسته ☐

شیوه ممیزی	
منابع رویدادنگاری	
شیوه یا ابزار تحلیل	
امکان ترمیم	

۲-۳ رهنمون‌های فرآیند جرم‌شناسی

در این بخش، برخی از مهم‌ترین رهنمون‌های مربوط به مراحل مختلف فرآیند جرم‌شناسی پایگاه‌داده آورده شده است. لازم به ذکر است که ملاحظات ذکر شده در این بخش برای هر سامانه پایگاه‌داده در هر سازمانی حیاتی است. با این وجود، ممکن است متناسب با نوع سازمان و نوع سامانه پایگاه‌داده، ملاحظات دیگری نیز افزوده شود [۱۲-۱۵].

۱. در صورت بروز نقض امنیتی، کل سازمان باید از آن مطلع شده و نشست آموزشی کوتاهی در زمینه رسیدگی به آن با حضور تمامی افراد برگزار شود. در صورتی که اطلاع‌رسانی به بخش‌های مختلف سازمان و تجزیه و تحلیل جرم‌شناسی توسط بخش‌های مختلف، طبق برنامه‌ی مشخصی انجام نشود، بخش‌هایی ممکن است موضوع را نادیده بگیرند و بخش‌های دیگری ممکن است آن را به اطلاع عموم برسانند. بنابراین، هرآنچه که در روند جرم‌شناسی انجام می‌شود می‌بایست طبق برنامه از پیش تدوین شده، صورت پذیرد.

۲. یکی از چالشی‌ترین مسائل در بررسی پایگاه‌های داده آن است که هر چه بیشتر در پایگاه‌داده جستجو و بررسی انجام شود، تغییرات بیشتری در آن رخ می‌دهد. لازم به ذکر است که تقریباً هر عملی که در پایگاه‌داده انجام می‌شود، می‌تواند باعث تغییر در رکوردهای دیکشنری شود.

۳. استفاده از حساب کاربری با مجوزهای بالا می‌تواند خود زمینه ایجاد تغییرات در پایگاه‌داده را فراهم آورد. بنابراین، در نظر گرفتن حساب کاربری با دسترسی‌های محدود فقط خواندنی برای این منظور ایده‌آل است. اگر چنین حساب‌های کاربری قبل از وقوع جرم تعریف نشده باشد، ایجاد حساب کاربری جدید توصیه نمی‌شود؛ چرا که خود موجب تغییرات جدیدی در سامانه می‌شود و با وجود اینکه استفاده از کاربر SYS می‌تواند خطرناک باشد ولی منطقی‌تر است.

۴. برای قابل قبول بودن شواهد در دادگاه قانونی به هنگام طرح دعوی، دو اصل اساسی باید رعایت شود: ۱): اولین اصل آن است که می‌بایست ثابت شود که اطلاعات و شواهد گردآوری شده به صورت قانونی به دست آمده است و ۲): دومین اصل نیز آن است که می‌بایست ثابت شود که به هنگام جمع‌آوری شواهد و اطلاعات تغییری در آنها اعمال نشده است. برای اثبات اصل اول، پیش از

جمع‌آوری شواهد باید قانونی بودن آن‌ها بررسی و تأیید شود. همچنین اصل دوم نیز با مستندسازی شواهد و اعمال اجرا شده بر روی آن‌ها قابل اثبات است. در حقیقت جمع‌آوری شواهد باید طی گام‌های از پیش تدوین شده‌ای صورت گیرد تا اطمینان حاصل شود که شواهد جمع‌آوری شده، تغییر نکرده‌اند. برای اثبات صحت داده‌ها می‌توان از روش مجموع مقابله‌ای[۶] استفاده کرد. این روش می‌تواند بر روی شواهد مختلف از جمله یک فایل یا کل دیسک سخت افزاری اعمال شود. با این روش می‌توان ثابت کرد که شواهد جمع‌آوری شده همان اطلاعاتی است که در ادامه از آن‌ها برای تجزیه و تحلیل استفاده می‌شود و بدون تغییر به دادگاه قابل ارائه است. توجه به این نکته حائز اهمیت است که برای استفاده از روش مجموع مقابله‌ای نباید به راحتی از بسته‌های موجود در پایگاه‌های داده استفاده شود. به عنوان مثال، در پایگاه‌داده اوراکل، نباید از بسته‌ی DBMS_SQLHASH استفاده شود چون ممکن است خود این بسته توسط مهاجم تغییر داده شده باشد. بنابراین در طول بررسی‌های جرم‌شناسی باید بتوان اعتبار ابزارها و عدم تغییر آن‌ها را ثابت کرد. مهاجم همچنین می‌تواند دیدهای موجود در پایگاه‌داده را برای مخفی نگه داشتن اعمال خود تغییر دهد. بنابراین باید یا از جدول‌های پایه در بررسی‌های جرم‌شناسی استفاده شود و یا پیش از استفاده از دیدها از صحت آن‌ها اطمینان حاصل گردد.

۵. یکی از مصنوعات مهم پایگاه‌داده برای تجزیه و تحلیل جرم‌شناسی، رویدادهای ثبت شده توسط ممیزی است. استفاده از ممیزی این اطمینان را ایجاد می‌کند که همیشه شواهدی برای استفاده در تحلیل جرم‌شناسی وجود دارد. می‌توان پایگاه‌داده را برای ثبت رویدادهای مشخص همچون ایجاد یک کاربر، تغییر رمز عبور و دسترسی به محتوای یک جدول تنظیم کرد. در صورتی که ممیزی فعال نباشد می‌توان از سایر قابلیت‌های رویدادنگاری در پایگاه‌های داده برای ثبت تغییرات استفاده کرد. با این وجود، به کارگیری ممیزی با جزئیات کافی و ثبت تمامی اعمال اجرا شده توسط مهاجم، تجزیه و تحلیل جرم‌شناسی را ساده‌تر می‌کند. تنظیمات مربوط به ممیزی می‌بایست بر اساس برنامه‌ای از قبل تعیین شده، مشخص گردد. در واقع بر اساس این برنامه، هرآنچه که نیاز به دانستن است، مشخص می‌گردد. در ادامه، برخی از امکان‌ها برای انجام تنظیمات ممیزی ارایه شده‌اند:
   - افرادی که به پایگاه‌داده وارد یا از آن خارج می‌شوند.

---

۶ Checksum

- افرادی که خود را به جای مدیر پایگاه‌داده نمایش می‌دهند.
- تلاش برای حمله‌ی تزریق SQL[۲۷]
- تغییر در پروفایل کاربر
- تغییر در ساختار پایگاه‌داده

یک راه حل جامع برای تهیه‌ی ممیزی باید شامل ممیزی از خود ممیزی نیز باشد. در صورتی که مهاجم تلاش به حذف یک رکورد ممیزی کند، باید این عمل ثبت شود. همچنین تلاش برای تغییر تنظیمات ممیزی نیز باید ثبت گردد. در صورتی که ممیزی در پایگاه‌داده فعال باشد، اولین گام شناسایی تنظیمات ممیزی است. پس از آن می‌توان به رویدادهای ممیزی و استفاده از آن‌ها در بررسی‌های جرم‌شناسی پرداخت.

۶. شواهد و اطلاعات مربوط به جرم‌شناسی اغلب در مکان‌های مختلفی از پایگاه‌داده وجود دارند. شناخت شواهد و اطلاعات مهم و اولویت‌بندی آنها از اهمیت بالایی برخوردار است. به غیر از اطلاعاتی که می‌توان از طریق اجرای پرسمان بر روی پایگاه‌داده‌ی تغییریافته به‌دست‌آورد، می‌توان از طریق ابزارهایی که توسط سامانه پایگاه‌داده استفاده می‌شوند؛ همچون نهان‌گاه طرح اجرایی[۲۸] و رویدادنگاری تراکنش‌ها[۲۹] نیز شواهدی را جمع‌آوری کرد. یک طرح اجرایی، کارآمدترین روش اجرای درخواست‌های داده است که در نهان‌گاه طرح اجرایی برای استفاده‌ی مجدد ذخیره می‌شوند. رویدادنگاری تراکنش‌ها در شناخت پرسمان‌های اجرا شده بر روی پایگاه‌داده و برای بررسی‌ها و تحقیقات مختلف، مفید است. سایر منابع شامل فایل‌هایی هستند که تاریخچه‌ی مربوط به پایگاه‌داده را ذخیره می‌کنند. برخی از این فایل‌ها به طور اختصاصی در پایگاه‌داده کاربرد دارند، همچون فایل رویدادنگاری پایگاه‌داده و فایل‌های داده در حالی که سایر فایل‌ها همچون رویدادنگاری سرور وب و رویدادنگاری رویدادهای سیستمی یک سیستم‌عامل به طورخاص به سرور پایگاه‌داده اختصاص داده نمی‌شوند. در هنگام تصمیم‌گیری درمورد اینکه کدام داده اول جمع‌آوری شود، مهم است که سطح بی‌ثباتی یک فایل در نظر گرفته شود.

---

۲۷ SQL Injection

۲۸ Execution plan cache

۲۹ Transaction logs

۷. یکی از بزرگترین مسائل در تجزیه و تحلیل جرم‌شناسی در پایگاه‌داده آن است که به صورت معمول پایگاه‌های داده، رویدادهای مربوط به دسترسی و خواندن محتوای جداول را ثبت نمی‌کنند ولی در تمامی مواقع، تغییرات در پایگاه‌داده همچون بروزرسانی، درج و حذف داده‌ها ثبت می‌شوند. گاهی نیاز است که شواهدی برای بررسی سرقت داده‌ها شناسایی شود. سرقت داده‌ها، ممکن است به نحوی باشد که داده‌ها را از پایگاه‌داده حذف نکند. بنابراین می‌بایست به دنبال شواهدی برای دسترسی به داده‌ها از طریق پایگاه‌داده بود. دسترسی به این گونه شواهد معمولاً پیچیده است مگر اینکه ممیزی فعال باشد. در حقیقت تنها روشی که می‌توان از آن به طور حتم برای اثبات دسترسی به داده مشخص استفاده کرد، فعال کردن ممیزی روی آن پیش از دسترسی است. به طور طبیعی، ثبت فعالیت خواندن داده‌ها ایده‌آل است هرچند همیشه انجام نمی‌شود. بنابراین باید بتوان در کنار آن، از راه‌های جایگزین نیز استفاده کرد. بدین منظور اصولاً از همبستگی[۳۰] میان شواهد استفاده می‌شود. به عنوان مثال، در پایگاه‌داده اوراکل، ممکن است شواهدی مبنی بر ورود مهاجم و ایجاد اتصال به پایگاه‌داده وجود داشته باشد. مهاجم ممکن است پرسمان SELECT را وارد کرده باشد و بهینه‌ساز[۳۱] اوراکل وارد عملی برای کامپایل دستور SQL شده باشد. همچنین در صورتی که حمله بلافاصله مورد بررسی قرار گرفته باشد، پرسمان SQL استفاده شده توسط مهاجم ممکن است در SGA وجود داشته باشد. در این شرایط، در صورتی که حمله از طریق سرور وب انجام شده باشد، پرسمان SQL ممکن است در فایل رویدادنگاری مربوط به برنامه‌ی کاربردی تحت وب موجود باشد. بنابراین در صورتی که ممیزی بر روی سیستم فعال نباشد، باید از همبستگی میان شواهد مختلف برای نتیجه‌گیری درمورد سرقت اطلاعات استفاده شود.

۸. برای جلوگیری از حجیم شدن فایل‌های رویدادنگاری و ممیزی می‌توان خط‌مشی‌هایی را برای بازنویسی[۳۲] رویدادها در این گونه از فایل‌ها اعمال کرد. به عنوان مثال می‌توان مشخص کرد که در صورتی که حجم فایل‌های رویدادنگاری به ۱۰۰ مگابایت رسید، اطلاعات جدید از ابتدای فایل بر روی اطلاعات قبلی نوشته شود یا در یک دوره هفت روزه، فایل‌های رویدادنگاری جدید ایجاد شوند.

---

۳۰ Correlation

۳۱ Optimizer

۳۲ Rotate

درصورتی که بر روی فایل‌های رویدادنگاری و ممیزی به صورت دوره‌ای اطلاعات ثبت شود، هنگام تشخیص رویداد غیرمجاز، باید ابتدا سرور پایگاه‌داده متوقف شود تا از اجرای اعمال و پرسمان‌های جدید بر روی سرور خودداری شود. سپس می‌توان از فایل‌های مورد نیاز، نسخه‌ی پشتیبان تهیه کرد تا تحلیل‌های آتی بر روی نسخه‌های پشتیبان انجام شود.

۹. در صورتی که خط‌مشی‌هایی برای تهیه‌ی پشتیبان از پایگاه‌داده به صورت دوره‌ای وجود داشته باشد، وضعیت مطلوبی که پایگاه‌داده پیش از این، در آن قرار داشته است در دسترس خواهد بود. در نتیجه، در صورت وقوع رویداد غیرمجاز با بازگرداندن فایل‌های پشتیبان می‌توان به حالت مطلوب پیش از رویداد غیرمجاز تغییر وضعیت داد.

۱۰. در صورت وجود فایل‌های رویدادنگاری و ممیزی بر روی سیستمی که در آن پایگاه‌داده وجود دارد، مدیر پایگاه‌داده یا کاربر مخرب می‌تواند به طور موقت تهیه‌ی ممیزی و رویدادنگاری را متوقف کرده و اعمال مورد نظر خود را اجرا کند و یا تغییراتی را به صورت دستی در فایل‌های ممیزی و رویدادنگاری ایجاد کند. در صورتی که چندین نسخه از داده‌های حساس در مکان‌های مختلف وجود داشته باشد، امکان از دست رفتن داده‌های حساس در صورت حذف آن‌ها از یک مکان، کاهش می‌یابد. همچنین مطلوب است که تمامی رویدادها به سیستم مرکزی ارسال شده و از آن‌ها به طور مرکزی حفاظت و نگهداری شود.

۱۱. معمولاً مهاجمین پس از حمله به پایگاه‌داده، سعی به حذف ردپای خود در فایل‌های رویدادنگاری و ممیزی می‌کنند. بنابراین محدود کردن دسترسی به این فایل‌ها و کپی‌برداری از آن‌ها و ذخیره‌سازی در سرور مرکزی از اهمیت زیادی برخوردار است. همچنین با تشخیص اعمال تغییر غیرمجاز در فایل‌های رویدادنگاری و ممیزی می‌توان متوجه شد که رویدادهای ثبت شده در این فایل‌ها فاقد اعتبار هستند. به عنوان نمونه، هنگام اجرای پرسمان بر روی فایل‌های رویدادنگاری redo log مربوط به پایگاه‌داده اوراکل، در صورتی که به صورت دستی تغییری در این فایل‌ها اعمال شده باشد، یکی از خطاهای موجود در شکل ۲ و شکل ۳ نشان داده می‌شود که نشان‌دهنده‌ی تغییر غیرمجاز در این فایل‌ها و عدم اعتبار اطلاعات ذخیره شده، است.

```
ORA-00308: cannot open archived log 'C:\Users\Desktop\oracle\oradata\orcl\REDO02.LOG'
ORA-27046: file size is not a multiple of logical block size
OSD-04012: file size mismatch (OS 209715713)
00308. 00000 -  "cannot open archived log '%s'"
*Cause:    The system cannot access a required archived redo log file.
*Action:   Check that the off line log exists, the storage device is
           online, and the archived file is in the correct location.
           Then attempt to continue recovery or restart the recovery
           session.
```

شکل ۲: خطای تغییر غیرمجاز در فایل رویدادنگاری

```
ORA-00368: checksum error in redo log block
ORA-00353: log corruption near block 132010 change 21987320 time 02/02/2018 10:11:30
ORA-00334: archived log: 'C:\USERS\DESKTOP\ORACLE\ORADATA\ORCL\REDO02.LOG'
00368. 00000 - "checksum error in redo log block"
*Cause:    The redo block indicated by the accompanying error, is not
           vaild. It has a checksum that does not match the block contents.
*Action:   Do recovery with a good version of the log or do time based
           recovery up to the indicated time. If this happens when archiving,
           archiving of the problem log can be skipped by clearing the log
           with the UNARCHIVED option. This must be followed by a backup of
           every datafile to insure recoverability of the database.
*Action:   Restore correct file or reset logs.
```

شکل ۳: خطای تغییر غیرمجاز در فایل رویدادنگاری

۲-۴ جمع‌بندی

در این فصل، ابتدا بستر مورد نیاز برای جرم‌شناسی پایگاه‌داده را تحت عنوان تمهیدات جرم‌شناسی به طور مشروح مورد بحث و بررسی قرار دادیم. سپس فرآیند جرم‌شناسی را از شناسایی جرم تا تهیه و ارائه مستندات به مدیریت سازمان و دادگاه‌قانونی برای طرح دعوی تشریح کردیم. همچنین به منظور ارائه مستندات مربوط به رویدادهای غیرمجاز کشف‌شده در سامانه، فرم استاندارد زیر ارایه گردید.

جدول ۲: تهیه‌ی مستند از فرآیند جرم‌شناسی پایگاه‌داده

عنوان رویداد			
وقوع جرم	عدم وقوع ☐	وقوع خصمانه ☐	وقوع ناخواسته ☐
شیوه ممیزی			
منابع رویدادنگاری			
شیوه یا ابزار تحلیل			
امکان ترمیم			

۳ درج، حذف و مشاهده‌ی غیرمجاز محتوای جداول

در این فصل به بحث و بررسی رویدادهای غیرمجازی که به صورت خصمانه یا ناخواسته توسط کاربر/برنامه‌کاربردی بر روی پایگاه‌داده اعمال می‌شوند و موجب درج سطر (های) جدید، ویرایش و حذف

سطر (های) موجود و همچنین مشاهده تمامی یا بخشی از جدول (ها) در پایگاه‌داده می‌شود، می‌پردازیم. رویدادهای فوق از جمله دستورات دستکاری داده (DML)[۳] در پایگاه‌داده به حساب می‌آیند. از آنجاییکه رویدادهای مورد بحث در این بخش از نقطه نظر جرم‌شناسی به هم نزدیک هستند، تنها بر روی رویداد حذف غیرمجاز به عنوان یک نماینده از این گروه رویدادها متمرکز شده و در صورت نیاز تفاوت‌های دیگر رویدادها ذکر می‌گردد.

۳-۱ شناسایی جرم

فرآیند جرم‌شناسی با مشاهده‌ی رویدادهای غیرمجاز در سامانه آغاز می‌شود. بنابراین در صورتیکه مدیر پایگاه‌داده خود به طور مستقیم یا غیر مستقیم (از طریق کاربران)، نسبت به حذف داده‌ها از جدولی آگاهی پیدا کند که انتظار حذف آنها در شرایط فعلی را نداشته است، فرآیند جرم‌شناسی در سامانه آغاز می‌شود.

۳-۲ جمع‌آوری اطلاعات و شواهد

از آنجاییکه هر پایگاه‌داده شواهد مربوط به اعمال مختلف را در بخش‌ها و فایل‌های رویدادنگاری مختلف ذخیره می‌کند، در این بخش قصد داریم منابع مربوط به شواهد رویداد غیرمجاز حذف را تعیین نماییم. در پایگاه‌داده‌ی MSSQL، با استفاده از رویدادنگاری در فایل ثبت تراکنش و همچنین ممیزی می‌توان اطلاعات مربوط به رویداد حذف داده از جدول را جمع‌آوری کرد. در ادامه به نحوه پیکربندی هر یک از این دو رویکرد می‌پردازیم.

رویدادنگاری در فایل ثبت تراکنش: هر پایگاه‌داده در یک نمونه از SQL Server دارای یک فایل ثبت است که تمامی تغییرات پایگاه‌داده در آن ثبت می‌شوند. به دلیل آنکه این نوع از فایل‌های ثبت به‌صورت مستقل پیش از اعمال تغییرات نوشته می‌شوند، در مواقع شکست[۴] در سخت‌افزار یا خطای برنامه‌ی کاربردی، امکان بازگرداندن[۵] یا برگشت به عقب[۶] تراکنش‌ها را فراهم می‌کنند. به دلیل اهمیت نقش فایل‌های ثبت تراکنش، رخدادها در یک یا چندین فایل ثبت به طور مجزا از فایل‌های داده، ذخیره می‌شوند.

---

۳ Data Manipulation Language

۴ ۳ Failure

۵ ۳ Restore

۶ ۳ Roll back

پیش از آنکه محتوای تغییر داده شده در حافظه‌ی بافر[۳۷] بر روی فایل‌های داده نوشته شوند، رکوردهای ثبت در فایل‌های ثبت تراکنش نوشته می‌شوند. به‌منظور به دست آوردن اطلاعاتی در مورد فایل ثبت تراکنش مربوط به یک پایگاه‌داده از دستور زیر استفاده می‌شود.

```
SELECT name,  size, -- in 8-KB pages  max_size, -- in 8-KB pages  growth,  is_percent_growth FROM
      sys.database_files  WHERE type_desc = 'LOG'
```

	name	size	max_size	growth	is_percent_growth
1	AdventureWorks2012_log	112	268435456	10	1

شکل ۴: فایل ثبت تراکنش مربوط به یک پایگاه‌داده

در شکل ۴ به دلیل آنکه پایگاه‌داده تنها دارای یک فایل ثبت تراکنش است، خروجی یک سطر است. برای به دست آوردن مسیر فایل ثبت تراکنش مربوط به یک پایگاه‌داده از دستور زیر استفاده می‌شود (شکل ۵).

```
SELECT * FROM sys.master_files WHERE name = 'AdventureWorks2012_log';
```

	›_desc	data_space_id	name	physical_name	state	state_desc
1	G	0	AdventureWorks2012_log	C:\Program Files (x86)\Microsoft SQL Server\MSSQL10_50.SADATABASE\MSSQL\DATA\AdventureWorks2012_log.ldf	0	ONLINE

شکل ۵: مسیر فایل ثبت تراکنش مربوط به یک پایگاه‌داده

اطلاعات تکمیلی: فایل ثبت تراکنش باید به طور منظم کوتاه‌سازی[۳۸] شود تا فضای دیسک را پر نکند. در صورتی که مدل بازیابی[۳۹] پایگاه‌داده، مدل ساده[۴۰] باشد، پس از هر checkpoint، کوتاه‌سازی به صورت خودکار انجام می‌شود. به منظور به تأخیر انداختن کوتاه‌سازی خودکار، می‌توان از مدل بازیابی کامل[۴۱] استفاده کرد. بدین ترتیب کوتاه‌سازی پس از پشتیبان‌گیری از فایل ثبت انجام می‌شود [۲]. به منظور آگاهی از مدل بازیابی یک پایگاه‌داده از دستور زیر استفاده می‌شود (شکل ۶):

---

۳۷ Buffer cache

۳۸ Truncate

۳۹ Recovery Model

۴۰ Simple Recovery Model

۴۱ Full Recovery Model

```
SELECT name, recovery_model_desc   FROM sys.databases  WHERE name = 'forensics' ;
```

	name	recovery_model_desc
1	forensics	FULL

شکل ۶: مدل بازیابی یک پایگاه‌داده

همچنین برای تغییر مدل بازیابی پایگاه‌داده به مدل بازیابی کامل دستور زیر وارد می‌شود:

```
ALTER DATABASE forensics SET RECOVERY FULL;
```

جدول زیر، گام‌های مورد نیاز برای جمع‌آوری اطلاعات و شواهد مربوط به رویداد حذف غیرمجاز محتوای جدول با استفاده از رویدادنگاری در فایل ثبت تراکنش را نشان می‌دهد.

یافتن محل فایل ثبت تراکنش		

۱	فهرست اسامی فایل(های) ثبت تراکنش مربوط به یک پایگاه‌داده	`SELECT name,  size, -- in 8-KB pages  max_size,`
		`-- in 8-KB  pages  growth,  is_percent_growth`
		`FROM  sys.database_files WHERE type_desc = 'LOG'`

۲	مسیر فایل ثبت تراکنش مربوط به یک پایگاه‌داده	`SELECT  *  FROM  sys.master_files` `WHERE name = '<FILE_NAME>';`
برخی دستورات سودمند		
۱	مدل بازیابی یک پایگاه‌داده	`SELECT name, recovery_model_desc  FROM sys.databases` `WHERE name = '<DATABASE_NAME>' ;`
۲	تغییر مدل بازیابی پایگاه‌داده به مدل بازیابی کامل	`ALTER DATABASE <DATABASE_NAME> SET RECOVERY` `FULL;`

ممیزی: ویژگی ممیزی در SQL Server امکان تهیه‌ی ممیزی از تمام اتفاقات رخ داده در سرور را فراهم می‌کند. رویدادها از تغییر در تنظیمات سرور تا تغییر در مقداری در جدول خاصی از پایگاه‌داده را شامل

می‌شوند. رکوردهای ممیزی در محل ثبت رویدادهای امنیتی ویندوز[۴۲]محل ثبت رویدادهای کاربردی ویندوز[۴۳]یا در فایل نوشته می‌شوند. در SQL Server 2012 امکان ممیزی در سطح سرور در تمامی نسخه‌ها وجود دارد ولی ممیزی پایگاه‌داده تنها در نسخه‌های Developer، Evaluation و Enterprise قابل اعمال است. آنچه در قسمت ممیزی بیان می‌شود، بر روی MSSQL Server 2012 نسخه‌ی Evaluation تهیه شده‌است. با استفاده از دستور زیر می‌توان لیست ممیزی‌های فعال تعریف شده را مشاهده کرد (شکل ۷).

```
SELECT      a.audit_id,
            a.name as audit_name,
            s.name as database_specification_name,
            d.audit_action_name,
            s.is_state_enabled,
            d.is_group,
            s.create_date,
            s.modify_date,
            d.audited_result
FROM sys.server_audits AS a JOIN sys.database_audit_specifications AS s ON a.audit_guid = s.audit_guid
JOIN sys.database_audit_specification_details AS d ON s.database_specification_id =
    d.database_specification_id    WHERE s.is_state_enabled = 1
```

audit_id	audit_name	database_specification_name	audit_action_name	is_state_enabled	is_group	create_date	modify_date	audited_result
65556	audit_delete	AW_DDL_Access_dbSpec	DELETE	1	0	2018-04-05 10:36:31.117	2018-04-05 10:36:31.117	SUCCESS AND FAILURE

شکل ۷: لیست ممیزی‌های فعال تعریف شده

همان‌طور که در شکل ۷ دیده می‌شود، مشخصه‌ی ممیزی برای تهیه‌ی ممیزی از عملیات delete در حالت موفقیت و عدم موفقیت تعریف و فعال شده است.

[۴] Windows security log

[۴] Windows application log

اطلاعات تکمیلی: قابلیت ممیزی در SQL Server شامل سه جزء اصلی است، ۱): ممیزی سرور، ۲): مشخصه‌ی ممیزی سرور و ۳): مشخصه‌ی ممیزی پایگاه‌داده. ممیزی سرور، سرآغازی برای تعریف مشخصه‌ی ممیزی در سرور SQL است.

```
CREATE SERVER AUDIT [audit_delete] TO FILE
(FILEPATH = 'C:\audit'
,MAXSIZE = 10 MB
)
WITH
(QUEUE_DELAY = 1000
,ON_FAILURE = CONTINUE
)
```

پس از تعریف ممیزی باید آن را فعال کرد تا مقصد ممیزی، داده‌ها را دریافت کند.

```
ALTER SERVER AUDIT [audit_delete] WITH (STATE = ON)
```

مشخصه‌ی ممیزی پایگاه‌داده، از رویدادها در سطح پایگاه‌داده ممیزی می‌گیرد. با استفاده از مشخصه‌ی ممیزی پایگاه‌داده، ممیزی می‌تواند در سطح شیئ یا کاربر انجام شود ولی در سطح ستون قابل انجام نیست.

```
CREATE DATABASE AUDIT SPECIFICATION [AW_DDL_Access_dbSpec]
FOR SERVER AUDIT [audit_delete]
ADD (DELETE ON DATABASE::test_forensics BY public)
WITH (STATE = ON)
```

جدول زیر، گام‌های مورد نیاز برای جمع‌آوری اطلاعات و شواهد مربوط به رویداد حذف غیرمجاز محتوای جدول با استفاده از ممیزی را نشان می‌دهد.

فهرست ممیزی‌های فعال تعریف شده در سطح پایگاه‌داده

```
SELECT      a.audit_id,
            a.name as audit_name,
            s.name as database_specification_name,
            d.audit_action_name,
            s.is_state_enabled,
            d.is_group,
            s.create_date,
            s.modify_date,
            d.audited_result
```

FROM sys.server_audits AS a JOIN sys.database_audit_specifications AS s ON a.audit_guid = s.audit_guid JOIN sys.database_audit_specification_details AS d ON s.database_specification_id = d.database_specification_id WHERE s.is_state_enabled = 1		
فعال کردن مشخصه‌ی ممیزی پایگاه‌داده		
CREATE SERVER AUDIT [audit_delete] TO FILE (FILEPATH = 'C:\audit' ,MAXSIZE = 10 MB) WITH (QUEUE_DELAY = 1000 ,ON_FAILURE = CONTINUE)	تعریف ممیزی سرور	۱
ALTER SERVER AUDIT [audit_delete] WITH (STATE = ON)	فعال‌سازی ممیزی سرور	۲
CREATE DATABASE AUDIT SPECIFICATION [DELETE_dbSpec] FOR SERVER AUDIT [audit_delete] ADD (DELETE ON DATABASE::test_forensics BY public) WITH (STATE = ON)	تعریف مشخصه‌ی ممیزی پایگاه‌داده	۳

۳-۳ استخراج و تجزیه و تحلیل اطلاعات

در این بخش نحوه‌ی استخراج اطلاعات برای هر یک از منابع، مورد بحث و بررسی قرار می‌گیرد. با استفاده از اطلاعات جمع‌آوری شده، تحلیل‌گر باید داده‌های حذف‌شده را بررسی و در صورت مشاهده‌ی رویداد حذف غیرمجاز، آن را به عنوان جرم تلقی نماید. در ادامه برای هر یک از منابع حاوی شواهد برای رویدادهای سامانه، نحوه استخراج و تجزیه و تحلیل شواهد تشریح می‌گردد.

فایل ثبت تراکنش: به منظور مشاهده‌ی محتوای فایل ثبت تراکنش از تابع fn_dblog() استفاده می‌شود. در صورتی که داده‌ای حذف شده باشد، ستون مربوط به Operation مقدار LOP_DELETE_ROWS را خواهد داشت. بنابراین از دستور زیر به منظور مشاهده‌ی اطلاعاتی در مورد سطرهای حذف شده، استفاده می‌شود. شکل ۸ خروجی این دستور را نشان می‌دهد.

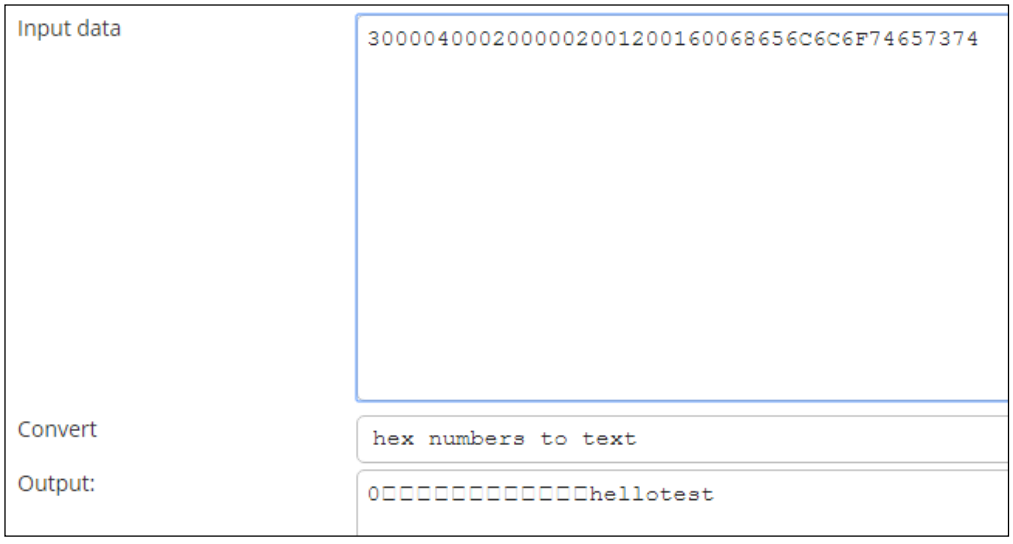
```
SELECT      [Transaction ID],
```

```
        Operation,
        Context,
        AllocUnitName,
        [Transaction Name],
        [Begin Time],
        [RowLog Contents 0]     FROM     fn_dblog(NULL, NULL)
        WHERE  Operation = 'LOP_DELETE_ROWS' OR  [Transaction Name] ='DELETE';
```

	Transaction ID	Operation	Context	AllocUnitName	Transaction Name	Begin Time	RowLog Contents 0
1	0000:000003a5	LOP_BEGIN_XACT	LCX_NULL	NULL	DELETE	2018/03/15 14:06:53:840	NULL
2	0000:000003a5	LOP_DELETE_ROWS	LCX_HEAP	dbo.test11	NULL	NULL	0x3000040002000002001200160068656C6C6F74657374
3	0000:000003a5	LOP_DELETE_ROWS	LCX_HEAP	dbo.test11	NULL	NULL	0x3000040002000002001200160068656C6C6F74657374
4	0000:000003a5	LOP_DELETE_ROWS	LCX_HEAP	dbo.test11	NULL	NULL	0x3000040002000002001200160068656C6C6F74657374
5	0000:000003a5	LOP_DELETE_ROWS	LCX_HEAP	dbo.test11	NULL	NULL	0x3000040002000002001200160068656C6C6F74657374
6	0000:000003a5	LOP_DELETE_ROWS	LCX_HEAP	dbo.test11	NULL	NULL	0x3000040002000002001200160068656C6C6F74657374
7	0000:000003a5	LOP_DELETE_ROWS	LCX_HEAP	dbo.test11	NULL	NULL	0x3000040002000002001200160068656C6C6F74657374
8	0000:000003a5	LOP_DELETE_ROWS	LCX_HEAP	dbo.test11	NULL	NULL	0x3000040002000002001200160068656C6C6F74657374
9	0000:000003a5	LOP_DELETE_ROWS	LCX_HEAP	dbo.test11	NULL	NULL	0x3000040002000002001200160068656C6C6F74657374

شکل ۸: مشاهده‌ی اطلاعاتی در مورد سطرهای حذف شده‌ی یک جدول

همان‌طور که در شکل ۸ دیده می‌شود، در تراکنش با شناسه‌ی یکتای 0000:000003a5 و با نام DELETE، هشت سطر حذف شده است و این رویداد در زمان گزارش شده در ستون Begin Time اتفاق افتاده است. این عملیات بر روی جدول test11 که در ستون AllocUnitName نشان داده شده است، اتفاق افتاده است. همچنین به منظور آگاهی از مقادیر موجود در سطرهای حذف شده، می‌توان مقدار هگزادسیمال موجود در ستون RowLog Contents 0 را به متن تبدیل کرد (شکل ۹).



شکل ۹: تبدیل مقدار هگزادسیمال به عدد

به منظور آگاهی از کاربر اجراکننده‌ی عمل حذف، از پرسمان زیر استفاده می‌شود. شکل ۱۰ خروجی حاصل از اجرای این پرسمان را نشان می‌دهد.

```
SELECT    [Current LSN],
          [Operation],
          [Transaction ID],
          [Description],
          SPID,
          [Begin Time],
          [Transaction SID],
          name 'LoginName'   FROM fn_dblog (NULL, NULL),
          (select sid,name from sys.syslogins )  sl
          WHERE [Transaction Name] LIKE '%delete%' and [Transaction SID] = sl.sid
```

	Current LSN	Operation	Transaction ID	Description	SPID	Begin Time	Transaction SID	LoginName
1	00000024:0000028b:0001	LOP_BEGIN_XACT	0000:000003a5	DELETE;0x01	54	2018/03/15 14:06:53:840	0x01	sa

شکل ۱۰: کاربر اجراکننده‌ی عمل حذف

همان‌طور که در شکل ۱۰ در ستون LoginName دیده می‌شود، عمل حذف توسط نام کاربری sa انجام شده‌است. جدول زیر، گام‌های مورد نیاز برای استخراج و تحلیل اطلاعات مربوط به رویداد حذف غیرمجاز محتوای جدول با استفاده از محتوای فایل ثبت تراکنش را نشان می‌دهد.

مشاهده‌ی محتوای فایل ثبت تراکنش و سطرهای حذف شده

```
SELECT      [Transaction ID],
            Operation,
            Context,
            AllocUnitName,
            [Transaction Name],
            [Begin Time],
            [RowLog Contents 0]    FROM    fn_dblog(NULL, NULL)
            WHERE      Operation = 'LOP_DELETE_ROWS' OR
                       [Transaction Name] = 'DELETE';
```

یافتن کاربر اجراکننده‌ی عمل حذف

```
SELECT      [Current LSN],
            [Operation],
            [Transaction ID],
```

```
[Description],
SPID,
[Begin Time],
[Transaction SID],
name 'LoginName'    FROM fn_dblog (NULL, NULL),
                    (select sid,name from sys.syslogins) sl
                    WHERE [Transaction Name] LIKE '%delete%' and
                    [Transaction SID] = sl.sid
```

ممیزی: اطلاعات ممیزی در فایل مقصد به صورت دودویی نوشته می‌شوند. به‌منظور خواندن فایل‌های دودویی ممیزی از تابع fn_get_audit_file() استفاده می‌شود (شکل ۱۱).

```
SELECT    event_time,
          session_server_principal_name,
          server_principal_name,
          database_name,
          object_name,
          statement          FROM      fn_get_audit_file('C:\audit\*', default, default);
```

	event_time	session_server_principal_name	server_principal_name	database_name	object_name	statement
1	2018-04-05 06:06:41.2183919	sa	sa	test_forensics	test	delete test where a = 1;
2	2018-04-05 06:06:42.1354444	sa	sa	test_forensics	test	delete test where a = 1;
3	2018-04-05 06:07:25.4119197	sa	sa	test_forensics	test	delete test where a = 1;
4	2018-04-05 06:07:54.9306080	sa	sa	test_forensics	test	delete test where a = 1;
5	2018-04-05 06:08:02.6120474	sa	sa	test_forensics	test	delete test where a = 1;

شکل ۱۱: خواندن فایل‌های دودویی ممیزی

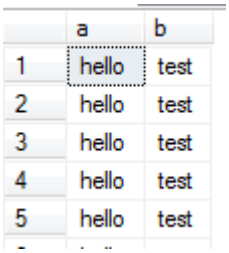
همان‌طور که در شکل ۱۱ دیده می‌شود، عبارت حذف اجرایی، کاربر و زمان اجرای آن بر روی شیئ مشخص با استفاده از اطلاعات ذخیره شده در ممیزی به دست می‌آید.

۳-۴ ترمیم

در صورتی که پیش از حذف داده‌ها اقدامات لازم جهت تهیه فایل پشتیبان صورت پذیرفته باشد، با بازیابی فایل می‌توان اطلاعات از دست رفته را مجدداً در اختیار گرفت. در این بخش سعی داریم روشی به غیر از روش‌های مربوط به تهیه‌ی نسخه‌های پشتیبان را برای بازیابی داده‌های از دست رفته مورد بحث و بررسی قرار دهیم.

با استفاده از رویه‌ی[۴] تعریف شده در پیوست الف می‌توان داده‌های حذف شده را بازیابی کرد. با اجرای این رویه با استفاده از دستور زیر، سطرهای حذف شده قابل بازیابی خواهند بود (شکل ۱۲).

```
EXEC Recover_Deleted_Data_Proc 'forensics','dbo.test11'
```



	a	b
1	hello	test
2	hello	test
3	hello	test
4	hello	test
5	hello	test

شکل ۱۲: بازیابی سطرهای حذف شده

ترمیم با اجرای رویه تعریف شده در [۱۶]
EXEC Recover_Deleted_Data_Proc '<DATABASE_NAME>','<SCHEMA_NAME>.<TABLE_NAME>'

۳-۵ ارائه‌ی مستندات

در این گام، اطلاعات کسب شده در طول فرآیند جرم‌شناسی پایگاه‌داده مستند می‌شود. برای این منظور می‌بایست برای هر یک از رویدادهای درج، حذف و مشاهده جداول در پایگاه‌داده به هنگام بررسی جرم، جداول زیر به طور دقیق تکمیل و به مدیریت سازمان برای پیگیری‌های لازم و طرح دعوی ارائه گردد.

درج غیرمجاز در جدول			
وقوع جرم	عدم وقوع ☐	وقوع خصمانه ☐	وقوع ناخواسته ☐
شیوه ممیزی	مشخصه‌ی ممیزی پایگاه‌داده ☐		

[۴] Stored Procedure

منابع رویدادنگاری	رویدادنگاری در فایل ثبت تراکنش ☐
شیوه یا ابزار تحلیل	فاقد شیوه یا ابزار تحلیل است.
امکان ترمیم	حذف سطر(های) درج شده ☐
توضیحات	

حذف غیرمجاز محتوای جدول			
وقوع جرم	عدم وقوع ☐	وقوع خصمانه ☐	وقوع ناخواسته ☐
شیوه ممیزی	مشخصه‌ی ممیزی پایگاه‌داده ☐		
منابع رویدادنگاری	رویدادنگاری در فایل ثبت تراکنش ☐		
شیوه یا ابزار تحلیل	فاقد شیوه یا ابزار تحلیل است.		
امکان ترمیم	استفاده از رویه‌های تعریف شده ☐		
توضیحات			

مشاهده غیرمجاز محتوای جدول			
وقوع جرم	عدم وقوع ☐	وقوع خصمانه ☐	وقوع ناخواسته ☐
شیوه ممیزی	مشخصه‌ی ممیزی پایگاه‌داده ☐		
منابع رویدادنگاری	رویدادنگاری در فایل ثبت تراکنش ☐		
شیوه یا ابزار تحلیل	فاقد شیوه یا ابزار تحلیل است.		
امکان ترمیم	فاقد امکان ترمیم است.		
توضیحات			

۳-۶ جمع‌بندی

در این فصل به بحث و بررسی رویدادهای غیرمجازی که به صورت خصمانه یا ناخواسته توسط کاربر/برنامه‌کاربردی بر روی پایگاه‌داده اعمال می‌شوند و موجب درج سطر (های) جدید، حذف سطر (های) موجود یا مشاهده تمامی یا بخشی از جدول (ها) در پایگاه می‌گردند، پرداختیم. برای این منظور، پس از

شناسایی جرم، دو رویکرد با نام‌های رویدادنگاری در فایل ثبت تراکنش و ممیزی پایگاه‌داده برای جمع‌آوری شواهد و اطلاعات مربوط به جرم معرفی کردیم. در ادامه نیز تنظیمات مربوط به فعال‌سازی، استخراج اطلاعات و شواهد، تحلیل شواهد، ترمیم و تهیه مستندات در این رابطه را مورد بحث و بررسی قرار دادیم.

۴ بروزرسانی غیرمجاز محتوای جداول

در این فصل به بحث و بررسی رویدادهای غیرمجازی که به صورت خصمانه یا ناخواسته توسط کاربر/برنامه‌کاربردی بر روی پایگاه‌داده اعمال و موجب بروزرسانی و تغییر داده‌های پایگاه‌داده می‌شوند، می‌پردازیم.

۴-۱ شناسایی جرم

فرآیند جرم‌شناسی با مشاهده‌ی رویدادهای غیرمجاز در سامانه آغاز می‌شود. بنابراین در صورتی که مدیر پایگاه‌داده خود به طور مستقیم یا غیرمستقیم (از طریق کاربران)، نسبت به تغییر و دستکاری برخی از داده‌های جدولی آگاهی پیدا کند که انتظار تغییر آنها در شرایط فعلی را نداشته است، فرآیند جرم‌شناسی در سامانه آغاز می‌شود.

۴-۲ جمع‌آوری اطلاعات و شواهد

از آنجایی که هر پایگاه‌داده شواهد مربوط به اعمال مختلف را در بخش‌ها و فایل‌های رویدادنگاری مختلف ذخیره می‌کند، در این بخش قصد داریم منابع مربوط به شواهد رویداد بروزرسانی غیرمجاز را تعیین نماییم. از این‌رو، در ادامه به معرفی منابع مربوط به شواهد رویداد بروزرسانی غیرمجاز و نحوه آگاهی و تنظیم آنها می‌پردازیم.

رویدادنگاری در فایل ثبت تراکنش: هر پایگاه‌داده در یک نمونه از SQL Server، دارای یک فایل ثبت است که تمامی تغییرات پایگاه‌داده در آن ثبت می‌شوند. به دلیل آنکه این نوع از فایل‌های ثبت به صورت مستقل پیش از اعمال تغییرات نوشته می‌شوند، در مواقع شکست[۵] سخت‌افزار یا خطای برنامه‌ی کاربردی، امکان بازگرداندن[۶] یا برگشت به عقب[۷] تراکنش‌ها را فراهم می‌کنند. به دلیل اهمیت نقش

---

۵ Failure

۶ Restore

فایل‌های ثبت تراکنش، رخدادها در یک یا چندین فایل ثبت به صورت مجزا از فایل‌های داده ذخیره می‌شوند. رکوردهای ثبت قبل از آنکه محتوای تغییر داده شده در حافظه‌ی بافر[۴۸] بر روی فایل‌های داده نوشته شوند، در فایل‌های ثبت تراکنش نوشته می‌شوند. به منظور به دست آوردن اطلاعاتی در مورد فایل ثبت تراکنش مربوط به یک پایگاه‌داده از دستور زیر استفاده می‌شود.

```
SELECT      name,
            size,
            -- in 8-KB pages  max_size,
            -- in 8-KB pages  growth,
            is_percent_growth    FROM sys.database_files   WHERE type_desc = 'LOG'
```

برای به دست آوردن مسیر فایل ثبت تراکنش مربوط به یک پایگاه‌داده از دستور زیر استفاده می‌شود.

```
SELECT * FROM sys.master_files WHERE name = 'AdventureWorks2012_log';
```

جدول زیر، گام‌های مربوط به جمع‌آوری اطلاعات و شواهد با استفاده از رویدادنگاری در فایل ثبت تراکنش برای رویداد غیرمجاز بروزرسانی محتوای جدول را نشان می‌دهد.

یافتن محل فایل ثبت تراکنش		
SELECT name, size, -- in 8-KB pages max_size, -- in 8-KB pages growth, is_percent_growth FROM sys.database_files WHERE type_desc = 'LOG'	فهرست اسامی فایل(های) ثبت تراکنش مربوط به یک پایگاه‌داده	۱
SELECT * FROM sys.master_files WHERE name = '<FILE_NAME>';	مسیر فایل ثبت تراکنش	۲

[۴] Roll back [۷]

[۴] Buffer cache [۸]

	مربوط به یک پایگاه‌داده	
برخی دستورات سودمند		
۱	مدل بازیابی یک پایگاه‌داده	SELECT name, recovery_model_desc FROM sys.databases WHERE name = '<DATABASE_NAME>' ;
۲	تغییر مدل بازیابی پایگاه‌داده به مدل بازیابی کامل	ALTER DATABASE <DATABASE_NAME> SET RECOVERY FULL;

ممیزی: ویژگی ممیزی در SQL Server امکان تهیه‌ی ممیزی از تمام اتفاقات رخ داده در سرور را فراهم می‌کند. رویدادها از تغییرات در تنظیمات سرور تا تغییر در مقداری در جدول خاصی از پایگاه‌داده را شامل می‌شوند. رکوردهای ممیزی در محل ثبت رویدادهای امنیتی ویندوز[۴] محل ثبت رویدادهای کاربردی ویندوز[۵] یا در فایل جداگانه‌ای نوشته می‌شوند. در SQL Server 2012 امکان ممیزی در سطح سرور در تمامی نسخه‌ها وجود دارد ولی ممیزی پایگاه‌داده تنها در نسخه‌های Developer، Evaluation و Enterprise قابل اعمال است. آنچه در قسمت ممیزی بیان می‌شود، بر روی MSSQL Server 2012 نسخه‌ی Evaluation بررسی شده است. با استفاده از دستور زیر می‌توان لیست ممیزی‌های فعال تعریف شده را مشاهده کرد.

```
SELECT      a.audit_id,
            a.name as audit_name,
            s.name as database_specification_name,
            d.audit_action_name,
            s.is_state_enabled,
            d.is_group,
            s.create_date,
            s.modify_date,
            d.audited_result    FROM sys.server_audits AS a JOIN sys.database_audit_specifications
```

[۴] Windows security log

[۵] Windows application log

```
AS s ON a.audit_guid = s.audit_guid JOIN
sys.database_audit_specification_details AS d ON
s.database_specification_id = d.database_specification_id
WHERE s.is_state_enabled = 1
```

جدول زیر، گام‌های مربوط به جمع‌آوری اطلاعات و شواهد با استفاده از ممیزی برای رویداد غیرمجاز بروزرسانی محتوای جدول را نشان می‌دهد.

فهرست ممیزی‌های فعال تعریف شده در سطح پایگاه‌داده

```
SELECT a.audit_id,
a.name as audit_name,
s.name as database_specification_name,
d.audit_action_name,
s.is_state_enabled,
d.is_group,
s.create_date,
s.modify_date,
d.audited_result FROM sys.server_audits AS a JOIN
sys.database_audit_specifications AS s ON
a.audit_guid = s.audit_guid JOIN
sys.database_audit_specification_details AS d ON
s.database_specification_id = d.database_specification_id
WHERE s.is_state_enabled = 1
```

فعال کردن مشخصه‌ی ممیزی پایگاه‌داده

۱	تعریف ممیزی سرور	CREATE SERVER AUDIT [audit_update] TO FILE (FILEPATH = 'C:\audit' ,MAXSIZE = 10 MB) WITH (QUEUE_DELAY = 1000 ,ON_FAILURE = CONTINUE)
۲	فعال‌سازی ممیزی سرور	ALTER SERVER AUDIT [audit_update] WITH (STATE = ON)
۳	تعریف مشخصه‌ی ممیزی	CREATE DATABASE AUDIT SPECIFICATION

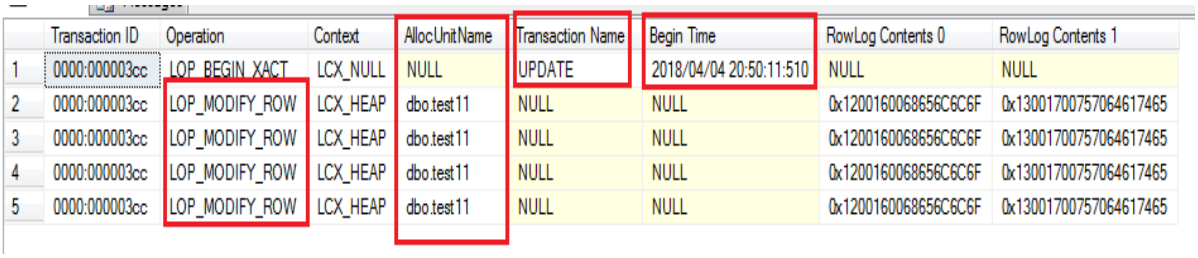
	پایگاه‌داده	[UPDATE_dbSpec] FOR SERVER AUDIT [audit_update] ADD (UPDATE ON DATABASE::test_forensics BY public) WITH (STATE = ON)

۴-۳ استخراج و تجزیه و تحلیل اطلاعات

در این بخش نحوه‌ی استخراج اطلاعات برای هر یک از منابع مربوط به شواهد اطلاعات مورد بحث و بررسی قرار می‌گیرد. با استفاده از اطلاعات جمع‌آوری‌شده، تحلیل‌گر باید داده‌های بروزرسانی شده را بررسی و در صورت مشاهده‌ی رویداد غیرمجاز، آن را به عنوان جرم/عیب تلقی نماید. در ادامه برای هر یک از منابع، نحوه استخراج و تجزیه و تحلیل شواهد بررسی می‌گردد.

استخراج و تحلیل اطلاعات موجود در فایل ثبت تراکنش: به منظور مشاهده‌ی محتوای فایل ثبت تراکنش از تابع fn_dblog() استفاده می‌شود. در صورتی که داده‌ای به‌روزرسانی شده باشد، ستون مربوط به Operation مقدار LOP_MODIFY_ROW را خواهد داشت. بنابراین از دستور زیر به منظور مشاهده‌ی اطلاعاتی در مورد سطرهای به‌روزرسانی شده، استفاده می‌شود (شکل ۱۳).

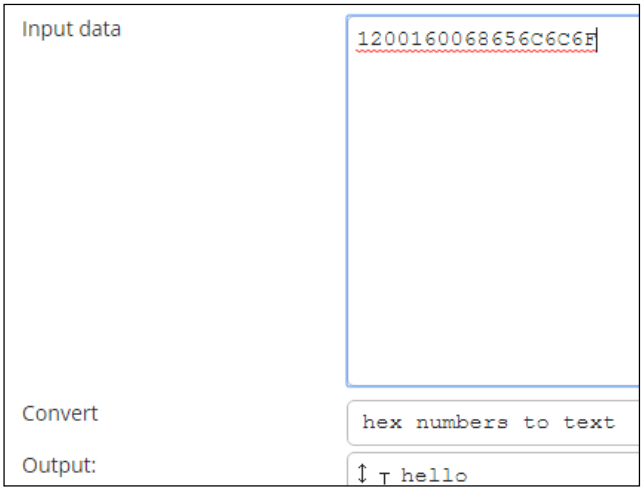
```
SELECT    [Transaction ID],
          Operation,
          Context,
          AllocUnitName,
        [Transaction Name],
        [Begin Time],
        [RowLog Contents 0],
        [RowLog Contents 1]    FROM    fn_dblog(NULL, NULL)
                                  WHERE   Operation = 'LOP_MODIFY_ROW' OR
                                          [Transaction Name] = 'UPDATE';
```



	Transaction ID	Operation	Context	AllocUnitName	Transaction Name	Begin Time	RowLog Contents 0	RowLog Contents 1
1	0000:000003cc	LOP_BEGIN_XACT	LCX_NULL	NULL	UPDATE	2018/04/04 20:50:11:510	NULL	NULL
2	0000:000003cc	LOP_MODIFY_ROW	LCX_HEAP	dbo.test11	NULL	NULL	0x1200160068656C6C6F	0x13001700757064617465
3	0000:000003cc	LOP_MODIFY_ROW	LCX_HEAP	dbo.test11	NULL	NULL	0x1200160068656C6C6F	0x13001700757064617465
4	0000:000003cc	LOP_MODIFY_ROW	LCX_HEAP	dbo.test11	NULL	NULL	0x1200160068656C6C6F	0x13001700757064617465
5	0000:000003cc	LOP_MODIFY_ROW	LCX_HEAP	dbo.test11	NULL	NULL	0x1200160068656C6C6F	0x13001700757064617465

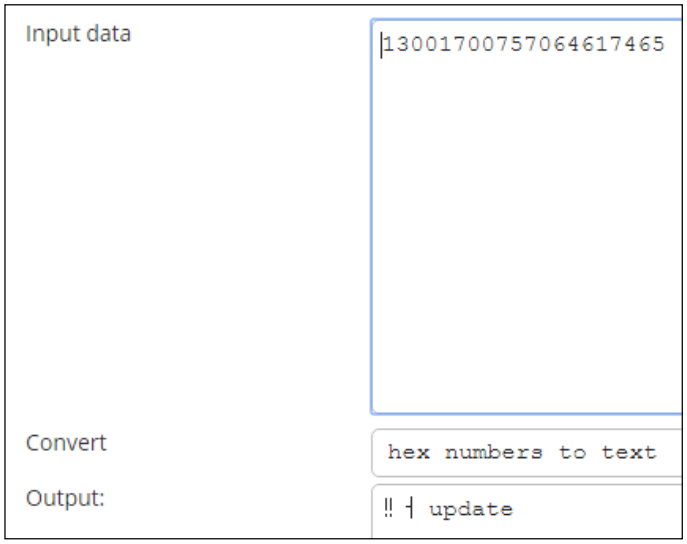
شکل ۱۳: مشاهده اطلاعات در مورد سطرهای به‌روزرسانی شده از یک جدول

همان‌طور که در شکل ۱۳ دیده می‌شود، در تراکنش با شناسه‌ی یکتای 0000.000003cc و با نام UPDATE، چهار سطر به‌روزرسانی شده‌اند و این رویداد در زمان موجود در ستون Begin Time اتفاق افتاده‌است. این عملیات بر روی جدول test11 که در ستون AllocUnitName نشان داده‌شده‌است، اتفاق افتاده‌است. همچنین به منظور آگاهی از مقدار فعلی در سطرهای بروزرسانی شده می‌توان مقدار هگزادسیمال موجود در ستون RowLog Contents 0 را به متن تبدیل کرد (شکل ۱۴).



شکل ۱۴: تبدیل مقدار هگزادسیمال به مقدار متنی

علاوه بر این، با تبدیل مقادیر موجود در ستون RowLog Contents 1 به متن، می‌توان مقادیر جدید جایگزین شده را به دست آورد (شکل ۱۵).



شکل ۱۵: تبدیل مقدار هگزادسیمال به مقدار متنی

به منظور آگاهی از کاربر اجراکننده‌ی عمل به‌روزرسانی، از پرسمان زیر استفاده می‌شود (شکل ۱۶).

```
SELECT    [Current LSN],
          [Operation],
          [Transaction ID],
          [Description], SPID, [Begin Time], [Transaction SID],
          name 'LoginName'   FROM fn_dblog (NULL, NULL), (select sid,name from sys.syslogins) sl
                             WHERE [Transaction Name] LIKE '%update%' and
                                   [Transaction SID] = sl.sid
```

	Current LSN	Operation	Transaction ID	Description	SPID	Begin Time	Transaction SID	LoginName
1	00000026:000001a6:0001	LOP_BEGIN_XACT	0000:000003cc	UPDATE;0x01	54	2018/04/04 20:50:11:510	0x01	sa

شکل ۱۶: کاربر اجراکننده‌ی عمل به‌روزرسانی

همان‌طور که در شکل ۱۶ در ستون LoginName دیده می‌شود، عمل به‌روزرسانی توسط نام کاربری sa انجام شده است. جدول زیر، گام‌های مربوط به استخراج و تجزیه و تحلیل اطلاعات برای رویداد غیرمجاز بروزرسانی محتوای جدول با استفاده از رویدادنگاری در فایل ثبت تراکنش را نشان می‌دهد.

مشاهده‌ی محتوای فایل ثبت تراکنش و سطرهای به‌روزرسانی شده

```
SELECT    [Transaction ID],
          Operation,
          Context,
          AllocUnitName,
          [Transaction Name],
          [Begin Time],
          [RowLog Contents 0],
          [RowLog Contents 1]    FROM     fn_dblog(NULL, NULL)
                                 WHERE    Operation = 'LOP_MODIFY_ROW' OR
                                          [Transaction Name] = 'UPDATE';
```

یافتن کاربر

```
SELECT    [Current LSN],
```

```
[Operation],
[Transaction ID],
[Description],
SPID,
[Begin Time],
[Transaction SID],
name 'LoginName'    FROM fn_dblog (NULL, NULL), (select sid,name from
                    sys.syslogins) sl WHERE [Transaction Name] LIKE
                    '%update%' and [Transaction SID] = sl.sid
```

استخراج و تحلیل اطلاعات با استفاده از ممیزی: اطلاعات ممیزی در فایل مقصد به‌صورت دودویی نوشته می‌شوند. به‌منظور خواندن فایل‌های دودویی ممیزی از تابع fn_get_audit_file() استفاده می‌شود.

```
SELECT    event_time,
          session_server_principal_name,
          server_principal_name,
          database_name,
          object_name,
          statement    FROM    fn_get_audit_file('C:\audit\*', default, default);
```

جدول زیر، گام مربوط به استخراج و تجزیه و تحلیل اطلاعات برای رویداد غیرمجاز بروزرسانی محتوای جدول با استفاده از ممیزی را نشان می‌دهد.

مشاهده‌ی محتوای فایل‌های دودویی ممیزی

```
SELECT    event_time,
          session_server_principal_name,
          server_principal_name,
          database_name,
          object_name,
          statement    FROM    fn_get_audit_file('C:\audit\*', default, default);
```

۴-۴ ترمیم

با استفاده از رویه تعریف شده در پیوست ب، می‌توان داده‌های به‌روزرسانی شده را بازیابی کرد. با اجرای این رویه با استفاده از دستور زیر، مقادیر قبلی در ستون‌های به‌روزرسانی شده قابل بازیابی خواهند بود.

```
EXEC Recover_Modified_Data_Proc 'forensics','dbo.test11'
```

	a	b	Update Statement
1	hello	test	UPDATE dbo.test11 SET [a]='hello' WHERE [b]='test'
2	hello	test	UPDATE dbo.test11 SET [a]='hello' WHERE [b]='test'
3	hello	test	UPDATE dbo.test11 SET [a]='hello' WHERE [b]='test'
4	hello	test	UPDATE dbo.test11 SET [a]='hello' WHERE [b]='test'

شکل ۱۷: بازیابی مقادیر به‌روزرسانی شده در یک جدول

همان‌طور که در شکل ۱۷ دیده می‌شود، مقادیر قبلی موجود در سطرهای به‌روزرسانی شده به همراه پرسمان‌هایی برای بازگرداندن مقدارهای به‌روزرسانی شده به مقادیر پیشین با اجرای رویه به دست می‌آید.

ترمیم با اجرای رویه تعریف شده در پیوست ب
EXEC Recover_Modified_Data_Proc <DATABASE_NAME>','<SCHEMA_NAME>.<TABLE_NAME>'

۴-۵ ارائه‌ی مستندات

در این گام، اطلاعات کسب شده در طول فرآیند جرم‌شناسی پایگاه‌داده مستند می‌شود. برای این منظور می‌بایست برای رویداد بروزرسانی داده‌های جداول به هنگام بررسی جرم، جدول زیر به طور دقیق تکمیل و به مدیریت سازمان برای پیگیری‌های لازم و طرح دعوی ارائه گردد.

جدول ۳: تهیه‌ی مستند از فرآیند جرم‌شناسی رویداد بروزرسانی غیرمجاز محتوای جدول

بروزرسانی غیرمجاز محتوای جدول			
وقوع جرم	عدم وقوع ☐	وقوع خصمانه ☐	وقوع ناخواسته ☐
شیوه ممیزی	مشخصه‌ی ممیزی پایگاه‌داده ☐		
منابع رویدادنگاری	رویدادنگاری در فایل ثبت تراکنش ☐		
شیوه یا ابزار تحلیل	فاقد شیوه یا ابزار تحلیل است.		
امکان ترمیم	استفاده از رویه‌های تعریف شده ☐		

توضیحات	

۴-۶ جمع‌بندی

در این فصل به بحث و بررسی رویدادهای غیرمجازی که به صورت خصمانه یا ناخواسته توسط کاربر/برنامه‌کاربردی بر روی پایگاه‌داده اعمال می‌شوند و موجب بروزرسانی داده‌های جداول در پایگاه‌داده می‌گردند، پرداخته شد. برای این منظور، پس از شناسایی جرم، دو رویکرد به نام‌های رویدادنگاری در فایل ثبت تراکنش و ممیزی برای جمع‌آوری شواهد و اطلاعات مربوط به جرم معرفی گردید. در پایان نیز برای هر یک از این رویکردها، تنظیمات مربوط به فعال‌سازی، استخراج اطلاعات و شواهد، تحلیل شواهد، ترمیم و در پایان تهیه مستندات تشریح شد.

۵ تغییر غیرمجاز شمای پایگاه‌داده

در این فصل به بحث و بررسی رویدادهای غیرمجازی که به صورت خصمانه یا ناخواسته توسط کاربر/برنامه‌کاربردی بر روی پایگاه‌داده اعمال و موجب تغییر در شمای پایگاه‌داده می‌شود، می‌پردازیم. این دسته از رویدادها، از جمله دستورات تعریف داده (DDL)[۵] در پایگاه‌داده به حساب می‌آیند. از آنجاییکه رویدادهای مورد بحث در این بخش از نقطه نظر جرم‌شناسی بسیار به هم نزدیک هستند، لذا تنها بر روی رویداد حذف غیرمجاز یک جدول متمرکز می‌شویم.

۵-۱ شناسایی جرم

فرآیند جرم‌شناسی با مشاهده‌ی رویدادهای غیرمجاز در سامانه آغاز می‌شود. بنابراین در صورتیکه مدیر پایگاه‌داده خود به طور مستقیم یا غیرمستقیم (از طریق کاربران)، نسبت به تغییر در شمای پایگاه‌داده همچون حذف یک جدول آگاهی پیدا کند که انتظار تغییر آن در شرایط فعلی را نداشته است، فرآیند جرم‌شناسی در سامانه آغاز می‌شود.

[۵] Data Definition Language

۵-۲ جمع‌آوری اطلاعات و شواهد

از آنجاییکه هر پایگاه‌داده شواهد مربوط به اعمال مختلف را در بخش‌ها و فایل‌های رویدادنگاری مختلف ذخیره می‌کند، در این بخش قصد داریم منابع مربوط به شواهد رویداد حذف غیرمجاز یک جدول را تعیین نماییم. از این‌رو، در ادامه به معرفی منابع مربوط به شواهد رویداد حذف غیرمجاز یک جدول و نحوه آگاهی و تنظیم آن‌ها می‌پردازیم.

رویدادنگاری در فایل ثبت تراکنش: هر پایگاه‌داده در یک نمونه از SQL Server دارای یک فایل ثبت است که تمامی تغییرات پایگاه‌داده در آن ثبت می‌شوند. به دلیل آنکه این نوع از فایل‌های ثبت به‌صورت مستقل پیش از اعمال تغییرات نوشته می‌شوند، در مواقع شکست[۲] در سخت‌افزار یا خطای برنامه‌ی کاربردی، امکان بازگرداندن[۳] یا برگشت به عقب[۴] تراکنش‌ها را فراهم می‌کنند. به دلیل اهمیت نقش فایل‌های ثبت تراکنش، رخدادها در یک یا چندین فایل ثبت، مجزا از فایل‌های داده ذخیره می‌شوند. رکوردهای ثبت، پیش از آنکه محتوای تغییر داده شده در حافظه‌ی بافر[۵] روی فایل‌های داده نوشته شوند، در فایل‌های ثبت تراکنش نوشته می‌شوند. به‌منظور به دست آوردن اطلاعاتی در مورد فایل ثبت تراکنش مربوط به یک پایگاه‌داده از دستور زیر استفاده می‌شود.

```
SELECT      name,
            size, -- in 8-KB pages
            max_size, -- in 8-KB pages
            growth,
            is_percent_growth    FROM sys.database_files   WHERE type_desc = 'LOG'
```

برای به دست آوردن مسیر فایل ثبت تراکنش مربوط به یک پایگاه‌داده از دستور زیر استفاده می‌شود.

```
SELECT * FROM sys.master_files WHERE name = 'AdventureWorks2012_log';
```

---

۲ Failure ۵

۳ Restore ۵

۴ Roll back ۵

۵ Buffer cache ۵

جدول زیر، گام مربوط به جمع‌آوری شواهد و اطلاعات برای رویداد تغییر غیرمجاز شمای پایگاه‌داده با استفاده از رویدادنگاری در فایل ثبت تراکنش را نشان می‌دهد.

یافتن محل فایل ثبت تراکنش		
۱	فهرست اسامی فایل(های) ثبت تراکنش مربوط به یک پایگاه‌داده	SELECT name, size, -- in 8-KB pages max_size, -- in 8-KB pages growth, is_percent_growth FROM sys.database_files WHERE type_desc = 'LOG'
۲	مسیر فایل ثبت تراکنش مربوط به یک پایگاه‌داده	SELECT * FROM sys.master_files WHERE name = '<FILE_NAME>';
برخی دستورات سودمند		
۱	مدل بازیابی یک پایگاه‌داده	SELECT name, recovery_model_desc FROM sys.databases WHERE name = '<DATABASE_NAME>' ;
۲	تغییر مدل بازیابی پایگاه‌داده به مدل بازیابی کامل	ALTER DATABASE <DATABASE_NAME> SET RECOVERY FULL;

ممیزی: ویژگی ممیزی در SQL Server امکان تهیه‌ی ممیزی از تمام اتفاقات رخ داده در سرور را فراهم می‌کند. رویدادها از تغییرات در تنظیمات سرور تا تغییر در مقداری در جدول خاصی از پایگاه‌داده را شامل می‌شوند. رکوردهای ممیزی در محل ثبت رویدادهای امنیتی ویندوز[۵۶]محل ثبت رویدادهای کاربردی ویندوز[۵۷] یا در فایل دیگری نوشته می‌شوند. در SQL Server 2012 امکان ممیزی در سطح سرور در تمامی نسخه‌ها وجود دارد ولی ممیزی پایگاه‌داده تنها در نسخه‌های Developer، Evaluation و Enterprise قابل

[۵] Windows security log

[۵] Windows application log

اعمال است. آنچه در قسمت ممیزی بیان می‌شود، بر روی MSSQL Server 2012 نسخه‌ی Evaluation تهیه شده‌است. با استفاده از دستور زیر می‌توان لیست ممیزی‌های فعال تعریف شده در پایگاه داده را مشاهده کرد.

```
SELECT      a.audit_id,
            a.name as audit_name,
            s.name as database_specification_name,
            d.audit_action_name,
            s.is_state_enabled,
            d.is_group,
            s.create_date,
            s.modify_date,
            d.audited_result    FROM sys.server_audits AS a JOIN
                                sys.database_audit_specifications AS s ON
                                a.audit_guid = s.audit_guid JOIN
                                sys.database_audit_specification_details AS d ON
                                s.database_specification_id = d.database_specification_id
                                WHERE s.is_state_enabled = 1
```

	audit_id	audit_name	database_specification_name	audit_action_name	is_state_enabled	is_group	create_date	modify_...	audited_result
1	65556	audit_delete	AW_DDL_Access_dbSpec	DELETE	1	0	2018-04-05 12:38:08.043	2018-0...	SUCCESS AND FAILURE
2	65556	audit_delete	AW_DDL_Access_dbSpec	UPDATE	1	0	2018-04-05 12:38:08.043	2018-0...	SUCCESS AND FAILURE
3	65558	audit_ddl	ddl_auditing	DATABASE_OBJECT_CHANGE_GROUP	1	1	2018-04-07 21:56:56.633	2018-0...	SUCCESS AND FAILURE
4	65558	audit_ddl	ddl_auditing	SCHEMA_OBJECT_CHANGE_GROUP	1	1	2018-04-07 21:56:56.633	2018-0...	SUCCESS AND FAILURE

شکل ۱۸: لیست ممیزی‌های فعال تعریف شده

همان‌طور که در شکل ۱۸ دیده می‌شود، مشخصه‌ی ممیزی برای تهیه‌ی ممیزی از اعمال DDL در حالت موفقیت و عدم موفقیت، تعریف و فعال شده است.

اطلاعات تکمیلی: ممیزی سرور، سرآغازی برای تعریف مشخصه‌ی ممیزی در سرور SQL است.

```
CREATE SERVER AUDIT [audit_ddl] TO FILE
(FILEPATH = 'C:\audit'
,MAXSIZE = 10 MB
)
WITH
(QUEUE_DELAY = 1000
,ON_FAILURE = CONTINUE
)
```

پس از تعریف ممیزی باید آن را فعال کرد تا مقصد ممیزی، داده‌ها را دریافت کند.

```
ALTER SERVER AUDIT [audit_ddl] WITH (STATE = ON)
```

مشخصه‌ی ممیزی پایگاه‌داده، از رویدادها در سطح پایگاه‌داده ممیزی می‌گیرد. با استفاده از مشخصه‌ی ممیزی پایگاه‌داده، ممیزی می‌تواند در سطح شیئ یا کاربر انجام شود ولی در سطح ستون قابل انجام نیست.

```
CREATE DATABASE AUDIT SPECIFICATION [ddl_auditing]
FOR SERVER AUDIT [audit_ddl]
ADD (DATABASE_OBJECT_CHANGE_GROUP),
ADD (SCHEMA_OBJECT_CHANGE_GROUP)
WITH (STATE = ON)
```

جدول زیر، گام مربوط به جمع‌آوری شواهد و اطلاعات برای رویداد تغییر غیرمجاز شمای پایگاه‌داده با استفاده از ممیزی را نشان می‌دهد.

فهرست ممیزی‌های فعال تعریف شده در سطح پایگاه‌داده

```
SELECT      a.audit_id,
            a.name as audit_name,
            s.name as database_specification_name,
            d.audit_action_name,
            s.is_state_enabled,
            d.is_group,
            s.create_date,
            s.modify_date,
            d.audited_result    FROM
                                sys.server_audits AS a  JOIN
                                sys.database_audit_specifications AS s ON
                                a.audit_guid = s.audit_guid  JOIN
                                sys.database_audit_specification_details AS d ON
                                s.database_specification_id = d.database_specification_id
                               WHERE s.is_state_enabled = 1
```

فعال کردن مشخصه‌ی ممیزی پایگاه‌داده

۱	تعریف ممیزی سرور	CREATE SERVER AUDIT [audit_ddl] TO FILE (FILEPATH = 'C:\audit'

,MAXSIZE = 10 MB) WITH (QUEUE_DELAY = 1000 ,ON_FAILURE = CONTINUE)		
ALTER SERVER AUDIT [audit_ddl] WITH (STATE = ON)	فعال‌سازی ممیزی سرور	۲
CREATE DATABASE AUDIT SPECIFICATION [ddl_auditing] FOR SERVER AUDIT [audit_ddl] ADD (DATABASE_OBJECT_CHANGE_GROUP), ADD (SCHEMA_OBJECT_CHANGE_GROUP) WITH (STATE = ON)	تعریف مشخصه‌ی ممیزی پایگاه‌داده	۳

۵-۳ استخراج و تجزیه و تحلیل اطلاعات

در این بخش نحوه‌ی استخراج اطلاعات از منابع مربوط به شواهد، مورد بحث و بررسی قرار می‌گیرد. با استفاده از اطلاعات جمع‌آوری شده، تحلیل‌گر باید تغییرات در شمای پایگاه‌داده را بررسی و در صورت مشاهده‌ی رویداد غیرمجاز، آن را به عنوان جرم تلقی نماید. در ادامه برای هر یک از منابع حاوی شواهد برای رویدادهای سامانه، نحوه استخراج و تجزیه و تحلیل تشریح می‌گردد.

استخراج و تحلیل اطلاعات با استفاده از رویدادنگاری در فایل ثبت تراکنش: به‌منظور مشاهده‌ی محتوای فایل ثبت تراکنش از تابع fn_dblog() استفاده می‌شود. در صورتی که جدول یا شیئی حذف شده باشد، مقدار موجود در ستون Transaction Name برابر DROPOBJ خواهد بود.

```
SELECT    [Transaction ID],
          Operation,
          Context,
          AllocUnitName,
          [Transaction Name],
          [Begin Time]    FROM fn_dblog (NULL, NULL)
                          WHERE [Transaction Name] = N'DROPOBJ';
```

همان‌طور که در شکل ۱۹ دیده می‌شود، با استفاده از اطلاعات استخراج شده از پرسمان فوق، نمی‌توان متوجه شد که کدامیک از اشیای موجود در پایگاه‌داده حذف شده‌اند (مقدار موجود در ستون AllocUnitName خالی است).

	Transaction ID	Operation	Context	Alloc Unit Name	Transaction Name	Begin Time
1	0000:00000327	LOP_BEGIN_XACT	LCX_NULL	NULL	DROPOBJ	2018/04/07 20:57:07:980
2	0000:0000032c	LOP_BEGIN_XACT	LCX_NULL	NULL	DROPOBJ	2018/04/07 21:09:24:617
3	0000:00000330	LOP_BEGIN_XACT	LCX_NULL	NULL	DROPOBJ	2018/04/07 21:19:32:237
4	0000:00000335	LOP_BEGIN_XACT	LCX_NULL	NULL	DROPOBJ	2018/04/07 21:20:59:120
5	0000:0000033d	LOP_BEGIN_XACT	LCX_NULL	NULL	DROPOBJ	2018/04/07 21:23:38:190
6	0000:00000345	LOP_BEGIN_XACT	LCX_NULL	NULL	DROPOBJ	2018/04/07 21:27:13:717
7	0000:0000034b	LOP_BEGIN_XACT	LCX_NULL	NULL	DROPOBJ	2018/04/07 21:51:00:270
8	0000:0000034e	LOP_BEGIN_XACT	LCX_NULL	NULL	DROPOBJ	2018/04/07 21:51:23:637
9	0000:00000355	LOP_BEGIN_XACT	LCX_NULL	NULL	DROPOBJ	2018/04/07 21:52:48:600
10	0000:0000035a	LOP_BEGIN_XACT	LCX_NULL	NULL	DROPOBJ	2018/04/07 21:54:25:830
11	0000:0000035e	LOP_BEGIN_XACT	LCX_NULL	NULL	DROPOBJ	2018/04/07 21:55:26:413
12	0000:00000364	LOP_BEGIN_XACT	LCX_NULL	NULL	DROPOBJ	2018/04/07 21:57:57:247

شکل ۱۹: لیست اشیای حذف شده از پایگاه‌داده

به منظور آگاهی از شیئ حذف شده از رویه‌ی تعریف شده در پیوست ج استفاده می‌شود که خروجی آن مشابه شکل ۲۰ خواهد بود.

```
EXEC Recover_Dropped_Objects_Detail_Proc
```

	Schema Name	Object Name	Dropped Date & Time	Dropped By User Name
1	dbo	test	2018/04/07 21:35:03:523	dbo
2	dbo	test	2018/04/07 21:36:49:370	dbo
3	dbo	test	2018/04/07 21:42:15:297	dbo

شکل ۲۰: لیست اشیای حذف شده از پایگاه‌داده

همچنین به منظور آگاهی از کاربر حذف‌کننده‌ی جدول می‌توان از پرسمان زیر استفاده کرد (شکل ۲۱).

```
SELECT [Transaction Id], [Begin Time], SUSER_SNAME ([Transaction SID]) AS [User]
FROM fn_dblog (NULL, NULL)
WHERE [Transaction Name] = N'DROPOBJ';
```

	Transaction Id	Begin Time	User
1	0000:00000327	2018/04/07 20:57:07:980	sa
2	0000:0000032c	2018/04/07 21:09:24:617	sa
3	0000:00000330	2018/04/07 21:19:32:237	sa
4	0000:00000335	2018/04/07 21:20:59:120	sa
5	0000:0000033d	2018/04/07 21:23:38:190	sa
6	0000:00000345	2018/04/07 21:27:13:717	sa
7	0000:0000034b	2018/04/07 21:51:00:270	sa
8	0000:0000034e	2018/04/07 21:51:23:637	sa
9	0000:00000355	2018/04/07 21:52:48:600	sa
10	0000:0000035a	2018/04/07 21:54:25:830	sa
11	0000:0000035e	2018/04/07 21:55:26:413	sa
12	0000:00000364	2018/04/07 21:57:57:247	sa

شکل ۲۱: کاربر حذف‌کننده‌ی یک شی

جدول زیر، گام مربوط به استخراج و تجزیه و تحلیل اطلاعات برای رویداد تغییر غیرمجاز شمای پایگاه‌داده با استفاده از رویدادنگاری در فایل ثبت تراکنش را نشان می‌دهد.

مشاهده‌ی محتوای فایل ثبت تراکنش

```
SELECT    [Transaction ID],
          Operation,
          Context,
          AllocUnitName,
          [Transaction Name],
          [Begin Time]    FROM fn_dblog (NULL, NULL)
                          WHERE [Transaction Name] = N'DROPOBJ';
```

فهرست اشیای حذف شده (استفاده از رویه‌ی تعریف شده در پیوست ج)

```
EXEC Recover_Dropped_Objects_Detail_Proc
```

یافتن کاربر اجراکننده‌ی عمل حذف جدول

```
SELECT [Transaction Id], [Begin Time], SUSER_SNAME ([Transaction SID]) AS [User]
FROM fn_dblog (NULL, NULL)
WHERE [Transaction Name] = N'DROPOBJ';
```

استخراج و تحلیل اطلاعات با استفاده از ممیزی: اطلاعات ممیزی در فایل مقصد به‌صورت دودویی نوشته می‌شوند. به‌منظور خواندن فایل‌های دودویی ممیزی از تابع fn_get_audit_file() استفاده می‌شود.

```
SELECT    event_time,
          session_server_principal_name,
          server_principal_name,
          database_name,
          object_name,
          statement    FROM    fn_get_audit_file('C:\audit\*', default, default);
```

	event_time	session_server_principal_name	server_principal_name	database_name	object_name	statement
368	2018-04-07 17:05:03.4656439	sa	sa	master	dbo	create table test (a int, b int);
369	2018-04-07 17:05:03.4656439	sa	sa	master	test	create table test (a int, b int);
370	2018-04-07 17:05:03.5246473	sa	sa	master	test	drop table test;
371	2018-04-07 17:06:49.3687012	sa	sa	master	dbo	create table test (a int, b int);
372	2018-04-07 17:06:49.3697013	sa	sa	master	test	create table test (a int, b int);
373	2018-04-07 17:06:49.3717014	sa	sa	master	test	drop table test;
374	2018-04-07 17:12:15.2353398	sa	sa	master	dbo	create table test (a int, b int);
375	2018-04-07 17:12:15.2353398	sa	sa	master	test	create table test (a int, b int);
376	2018-04-07 17:12:15.2363398	sa	sa	master	test	alter table test add c int;
377	2018-04-07 17:12:15.2983434	sa	sa	master	test	drop table test;
378	2018-04-07 17:27:57.2122177	sa	sa	test_forensics	dbo	create table test (a int, b int);
379	2018-04-07 17:27:57.2122177	sa	sa	test_forensics	test	create table test (a int, b int);
380	2018-04-07 17:27:57.2462197	sa	sa	test_forensics	test	alter table test add c int;
381	2018-04-07 17:27:57.2482198	sa	sa	test_forensics	test	drop table test;

شکل ۲۲: مشاهده‌ی اطلاعاتی در مورد عبارات DDL اجرایی

همان‌طور که در شکل ۲۲ دیده می‌شود، عبارت DDL اجرایی، کاربر و زمان اجرای آن با استفاده از اطلاعات ذخیره شده در ممیزی به دست می‌آید.

مشاهده‌ی محتوای فایل‌های دودویی ممیزی

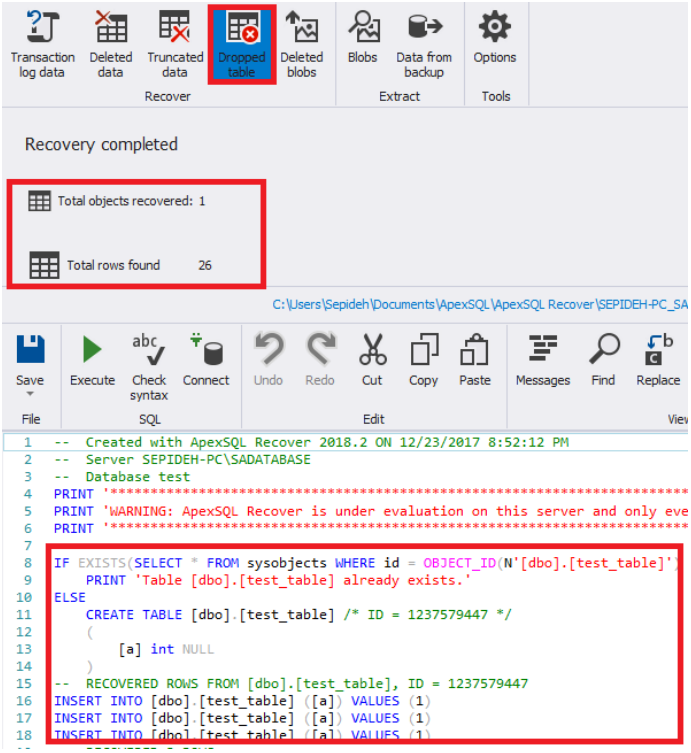
```
SELECT    event_time,
          session_server_principal_name,
          server_principal_name,
          database_name,
          object_name,
          statement    FROM    fn_get_audit_file('C:\audit\*', default, default);
```

۵-۴ ترمیم

در صورتی که پیش از تغییر در شمای پایگاه‌داده و به طور خاص حذف یک جدول، اقدامات لازم جهت تهیه فایل پشتیبان صورت پذیرفته باشد، با بازیابی فایل می‌توان اطلاعات از دست رفته را مجدداً در اختیار گرفت. همچنین در صورتی که از کلیه‌ی اعمال DML و DDL بر روی یک جدول از پایگاه‌داده ممیزی تهیه

شده‌باشد با استخراج پرسمان‌ها از زمان ایجاد جدول تا پیش از حذف آن و اجرای مجدد آن پرسمان‌ها می‌توان جدول و محتوای آن را بازیابی کرد.

در قسمت ترمیم در بخش‌های مختلف و برای رویدادهای ناخواسته‌ی متفاوت، سعی شده‌است که از قابلیت‌های موجود در پایگاه‌داده برای ترمیم رویداد ناخواسته استفاده شود ولی در این قسمت به دلیل عدم وجود راه‌حل کوتاه و خودکار توصیه می‌شود از ابزار ApexSQL Recover به منظور بازیابی داده‌های جدول حذف شده، استفاده شود. لازم به ذکر است در صورتی که عبارت DROP TABLE <table_name> اجرا شده باشد، استفاده از ابزارApexSQL Recover با احتمال زیادی بازیابی جدول و داده‌های درون آن را با موفقیت انجام می‌دهد (شکل ۲۳).



شکل ۲۳: استفاده از ابزار ApexSQL Recover به منظور بازیابی جدول حذف شده

اطلاعات تکمیلی: در این قسمت به بررسی مراحلی که بلافاصله پس از حذف داده‌ها به منظور کاهش احتمال نوشته شدن بر روی داده‌ها و از دست رفتن داده‌ها طی فرآیند بازیابی انجام می‌گردد، پرداخته

می‌شود.

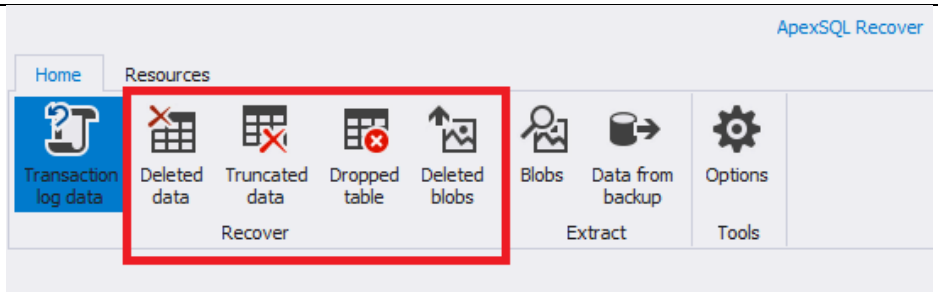
به منظور بازیابی داده‌های ازدست‌رفته، ابزار ApexSQL Recover فایل MDF و فایل‌های LDF را می‌خواند. اطلاعاتی در مورد رکوردهای کوتاه[۵۸] یا حذف شده[۵۹] تنها در فایل‌های MDF قابل دستیابی است. هنگامی‌که رکوردی حذف یا کوتاه می‌شود، برای نوشته شدن مجدد بر روی آن، در فایل MDF نشانه‌گذاری می‌شود. با انجام فعالیت‌های جدید بر روی پایگاه‌داده، این فعالیت‌ها بر روی رکوردها مجدداً نوشته می‌شوند و هنگامی‌که این اتفاق رخ دهد، ApexSQL Recover قابلیت بازیابی رکورد از دست رفته را نخواهد داشت. به همین دلیل جلوگیری از نوشته شدن مجدد بر روی فایل MDF از اهمیت بسیاری برخوردار است. بنابراین در صورت وقوع حادثه، بلافاصله باید گام‌های زیر طی شوند:

- پایگاه‌داده بلافاصله پس از حذف داده در وضعیت فقط خواندنی[۶۰] قرار داده شود: با این کار از نوشته شدن مجدد بر روی فایل MDF جلوگیری می‌شود و می‌توان از فایل‌های MDF و LDF نسخه‌برداری کرد.
- اجرای ApexSQL Recover بر روی پایگاه‌داده‌ای که در وضعیت فقط خواندنی قرار دارد یا نسخه‌برداری از فایل‌های MDF و LDF، بازگرداندن آن‌ها بر روی پایگاه‌داده‌ی جدید و اجرای ApexSQL Recover بر روی آن.
- در صورتی که چگونگی از دست رفتن داده مشخص باشد، باید گزینه‌ی مناسب به منظور بازیابی داده انتخاب شود (شکل ۲۴). در صورتی که نحوه‌ی از دست رفتن داده مشخص نباشد، هریک از گزینه‌ها تا بازیابی داده‌ها به صورت موفقیت‌آمیز، بررسی و اجرا می‌شوند. تا زمانی که داده‌های جدید بر روی داده‌های از دست رفته نوشته نشده‌اند، ابزار ApexSQL Recover داده‌ها را بازیابی خواهد کرد.

[۵] Truncated records

[۵] Dropped records

[۶] Read-only



شکل ۲۴: گزینه‌های بازیابی ApexSQL Recover

۵-۵ ارائه‌ی مستندات

در این گام، اطلاعات کسب شده در طول فرآیند جرم‌شناسی پایگاه‌داده مستند می‌شود. برای این منظور می‌بایست به هنگام بررسی جرم، جدول زیر به طور دقیق تکمیل و به مدیریت سازمان برای پیگیری‌های لازم و طرح دعوی ارائه گردد.

جدول ۴: تهیه‌ی مستند از فرآیند جرم‌شناسی رویداد تغییر غیرمجاز شمای پایگاه‌داده

تغییر غیرمجاز شمای پایگاه‌داده			
وقوع جرم	عدم وقوع ☐	وقوع خصمانه ☐	وقوع ناخواسته ☐
شیوه ممیزی	مشخصه‌ی ممیزی پایگاه‌داده ☐		
منابع رویدادنگاری	رویدادنگاری در فایل ثبت تراکنش ☐		
شیوه یا ابزار تحلیل	فاقد شیوه یا ابزار تحلیل است.		
امکان ترمیم	استفاده از ApexSQL Recover ☐		
توضیحات			

۵-۶ جمع‌بندی

در این فصل به بحث و بررسی رویدادهای غیرمجازی که به صورت خصمانه یا ناخواسته توسط کاربر/برنامه‌کاربردی بر روی پایگاه‌داده اعمال و موجب تغییر در شمای پایگاه‌داده می‌گردند، پرداخته شد. برای این منظور، پس از شناسایی جرم، دو رویکرد به نام‌های رویدادنگاری در فایل ثبت تراکنش و ممیزی برای جمع‌آوری شواهد و اطلاعات مربوط به جرم معرفی گردید. برای هر یک از این رویکردها، تنظیمات مربوط به فعال‌سازی، استخراج اطلاعات و شواهد، تحلیل شواهد، ترمیم و در پایان تهیه مستندات تشریح گردید.

۶ تلاش برای ورود غیرمجاز به پایگاه‌داده

در این فصل، رویدادهای غیرمجازی که به صورت خصمانه یا ناخواسته توسط کاربر/برنامه‌کاربردی برای ورود به پایگاه‌داده صورت می‌پذیرد را مورد بحث و بررسی قرار می‌دهیم. این رویداد در واقع به هنگام تلاش برای ورود از طریق بدست آوردن نام کاربری و کلمه عبور به سمپاد تحمیل می‌شود. در ادامه، فرآیند جرم‌شناسی مربوط به این رویداد مورد بحث و بررسی قرار می‌گیرد.

۶-۱ شناسایی جرم

در صورتی که تلاش‌های ناموفق برای ورود به سیستم پایگاه‌داده ثبت شوند، با مشاهده‌ی رویدادهای ثبت‌شده می‌توان حمله به پایگاه‌داده برای یافتن نام کاربری یا کلمه‌ی عبور یک نام کاربری را تشخیص داد. در این شرایط فرآیند جرم‌شناسی در سامانه آغاز می‌شود.

۶-۲ جمع‌آوری اطلاعات و شواهد

از آنجاییکه هر پایگاه‌داده شواهد مربوط به اعمال مختلف را در بخش‌ها و فایل‌های رویدادنگاری مختلف ذخیره می‌کند، در این بخش قصد داریم منابع مربوط به شواهد رویداد تلاش به منظور ورود به پایگاه‌داده را تعیین نماییم.

ویژگی ممیزی در SQL Server امکان تهیه‌ی ممیزی از تمام اتفاقات رخ داده در سرور را فراهم می‌کند. رویدادها از تغییر در تنظیمات سرور تا تغییر در مقداری در جدول خاصی از پایگاه‌داده را شامل می‌شوند. رکوردهای ممیزی در محل ثبت رویدادهای امنیتی ویندوز[۶۱] محل ثبت رویدادهای کاربردی ویندوز[۶۲] یا در فایل دیگری نوشته می‌شوند. در SQL Server 2012 امکان ممیزی در سطح سرور در تمامی نسخه‌ها وجود دارد ولی ممیزی پایگاه‌داده تنها در نسخه‌های Developer، Evaluation و Enterprise قابل اعمال است. آنچه در قسمت ممیزی بیان می‌شود، بر روی MSSQL Server 2012 نسخه‌ی Evaluation تهیه شده‌است. با استفاده از دستور زیر می‌توان لیست ممیزی‌های فعال تعریف شده در سطح سرور را مشاهده کرد.

```
SELECT      audit_id,
            a.name as audit_name,
```

[۶۱] Windows security log

[۶۲] Windows application log

```
s.name as server_specification_name,
d.audit_action_name,
s.is_state_enabled,
d.is_group,
d.audit_action_id,
s.create_date,
s.modify_date   FROM sys.server_audits AS a JOIN
                    sys.server_audit_specifications AS s ON
                    a.audit_guid = s.audit_guid JOIN
                    sys.server_audit_specification_details AS d ON
                    s.server_specification_id = d.server_specification_id
                    WHERE s.is_state_enabled = 1
```

	audit_id	audit_name	server_specification_name	audit_action_name	is_state_enabled	is_group
1	65559	failed_login	audit_failed_login	FAILED_LOGIN_GROUP	1	1

شکل ۲۵: لیست ممیزی‌های فعال تعریف شده در سطح سرور

همان‌طور که در شکل ۲۵ دیده می‌شود، مشخصه‌ی ممیزی برای تهیه‌ی ممیزی از ورود ناموفق تعریف و فعال شده است.

اطلاعات تکمیلی: ممیزی سرور، سرآغازی برای تعریف مشخصه‌ی ممیزی در سرور SQL است.

```
CREATE SERVER AUDIT [failed_login] TO FILE
(FILEPATH = 'C:\audit'
,MAXSIZE = 10 MB
)
WITH
(QUEUE_DELAY = 1000
,ON_FAILURE = CONTINUE
)
```

پس از تعریف ممیزی باید آن را فعال کرد تا مقصد ممیزی، داده‌ها را دریافت کند.

```
ALTER SERVER AUDIT [failed_login] WITH (STATE = ON)
```

مشخصات ممیزی سرور که در تمامی نسخه‌های SQL Server پشتیبانی می‌شود، برای تعریف ممیزی در سطح سرور مورداستفاده قرار می‌گیرد.

```
CREATE SERVER AUDIT SPECIFICATION [audit_failed_login]
FOR SERVER AUDIT [failed_login]
ADD (FAILED_LOGIN_GROUP)
WITH (STATE = ON)
```

جدول زیر، گام‌های مربوط به جمع‌آوری شواهد و اطلاعات با استفاده از ممیزی را برای رویداد ورود غیرمجاز به پایگاه‌داده نشان می‌دهد.

فهرست ممیزی‌های فعال تعریف شده در سطح سرور		
SELECT audit_id, a.name as audit_name, s.name as server_specification_name, d.audit_action_name, s.is_state_enabled, d.is_group, d.audit_action_id, s.create_date, s.modify_date FROM sys.server_audits AS a JOIN sys.server_audit_specifications AS s ON a.audit_guid = s.audit_guid JOIN sys.server_audit_specification_details AS d ON s.server_specification_id = d.server_specification_id WHERE s.is_state_enabled = 1		
فعال کردن مشخصه‌ی ممیزی سرور		
CREATE SERVER AUDIT [failed_login] TO FILE (FILEPATH = 'C:\audit' ,MAXSIZE = 10 MB) WITH (QUEUE_DELAY = 1000 ,ON_FAILURE = CONTINUE)	تعریف ممیزی سرور	۱
ALTER SERVER AUDIT [failed_login] WITH (STATE = ON)	فعال‌سازی ممیزی سرور	۲
CREATE SERVER AUDIT SPECIFICATION [audit_failed_login] FOR SERVER AUDIT [failed_login] ADD (FAILED_LOGIN_GROUP) WITH (STATE = ON)	تعریف مشخصه‌ی ممیزی سرور	۳

۶-۳ استخراج و تجزیه و تحلیل اطلاعات

با استفاده از اطلاعات جمع‌آوری شده، تحلیل‌گر می‌بایست تلاش‌های ناموفق برای ورود به پایگاه‌داده را بررسی و در صورت مشاهده‌ی تلاش‌های متوالی در فواصل زمانی کوتاه آن را به عنوان یک جرم تلقی نماید. استخراج و تجزیه و تحلیل اطلاعات برای رویداد ورود غیرمجاز به پایگاه‌داده را می‌توان از طریق رویدادهای ممیزی ثبت‌شده با استفاده از دستور زیر استخراج کرد.

```
SELECT      event_time,
            Statement   FROM     fn_get_audit_file('C:\audit\*', default, default)
            WHERE statement LIKE 'Login failed%';
```

همان‌طور که در شکل ۲۶ دیده می‌شود، در فواصل زمانی کوتاه، نام کاربری sa برای ورود تلاش کرده است و با شکست روبرو شده است. این رویدادهای ثبت شده نشان‌دهنده‌ی حمله‌ی brute force برای یافتن کلمه‌ی عبور مربوط به یک نام کاربری است. همچنین در فواصل زمانی کوتاه، نام‌های کاربری مختلف به منظور یافتن نام کاربری مجاز، امتحان شده‌اند.

	event_time	statement
1	2018-04-08 17:38:27.4855702	Login failed for user 'sa'. Reason: Password did not match that for the login provided. [CLIENT: <local machine>]
2	2018-04-08 17:38:32.3648493	Login failed for user 'sa'. Reason: Password did not match that for the login provided. [CLIENT: <local machine>]
3	2018-04-08 17:39:33.3283362	Login failed for user 'sa'. Reason: Password did not match that for the login provided. [CLIENT: <local machine>]
4	2018-04-08 17:39:36.0134898	Login failed for user 'sa'. Reason: Password did not match that for the login provided. [CLIENT: <local machine>]
5	2018-04-08 17:39:40.9617728	Login failed for user 'admin'. Reason: Could not find a login matching the name provided. [CLIENT: <local machine>]
6	2018-04-08 17:39:49.8022785	Login failed for user 'user'. Reason: Could not find a login matching the name provided. [CLIENT: <local machine>]

شکل ۲۶: تلاش برای یافتن کلمه‌ی عبور یک نام کاربری

مشاهده‌ی محتوای فایل‌های دودویی ممیزی

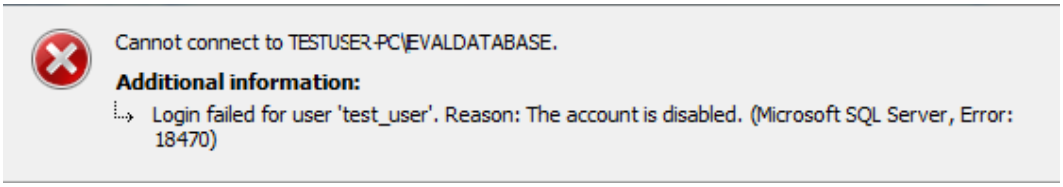
```
SELECT      event_time,
            Statement   FROM   fn_get_audit_file('C:\audit\*', default, default)
                        WHERE statement LIKE 'Login failed%';
```

۶-۴ ترمیم

در صورتی که رویدادهای ثبت شده نشان‌دهنده‌ی تلاش برای ورود به پایگاه‌داده باشند، بنابر صلاح دید مدیر پایگاه‌داده، می‌توان حساب کاربری را برای مدتی با دستور زیر غیرفعال کرد. بدین ترتیب به نوعی حمله‌ی حدس کلمه‌ی عبور دشوار می‌شود.

```
ALTER LOGIN test_user DISABLE
```

در صورتی که با استفاده از نام کاربری غیرفعال، تلاش برای ورود به پایگاه‌داده انجام شود، خطایی مشابه شکل ۲۷ تولید می‌شود.



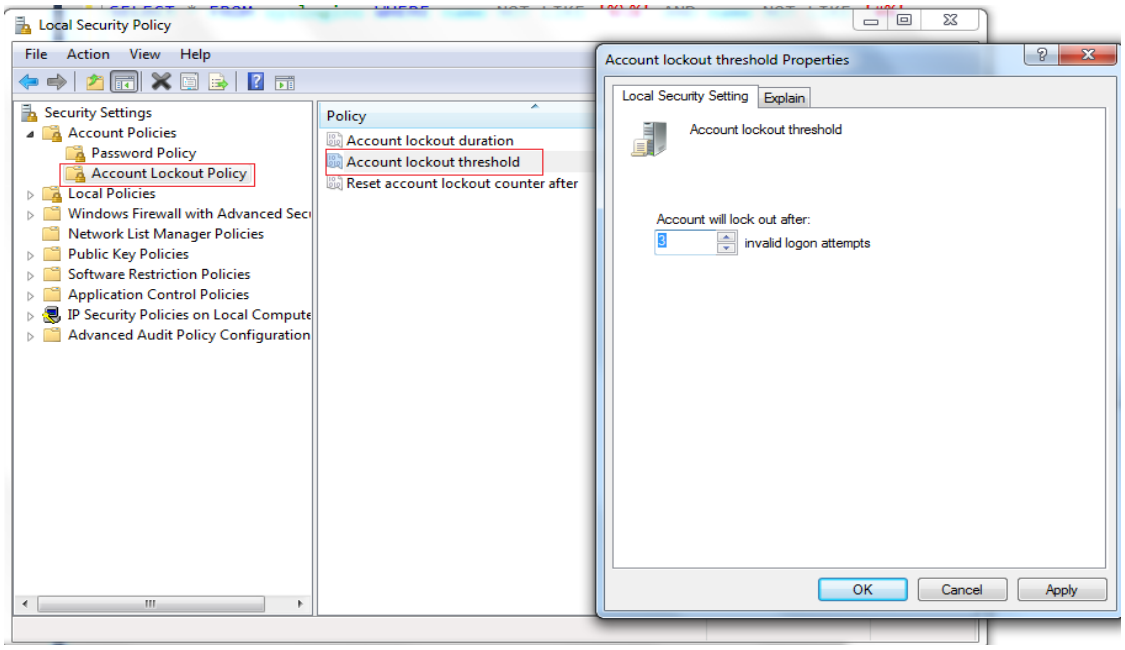
شکل ۲۷: خطای ناشی از ورود یک نام کاربری غیرفعال

همچنین با دستور زیر می‌توان نام کاربری را مجدداً فعال کرد.

```
ALTER LOGIN test_user ENABLE
```

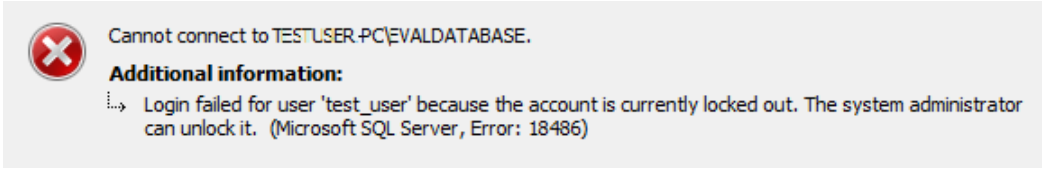
علاوه بر این می‌توان حساب کاربری را به صورت خودکار بعد از تعداد مشخصی تلاش ناموفق برای ورود، قفل کرد. برای تنظیم تعداد تلاش ناموفق برای ورود، ابتدا باید تنظیمات مربوطه در سیستم عامل ویندوز انجام شوند. بدین منظور از مسیر زیر، وارد قسمت Account lockout policy شده و خط‌مشی Account lockout threshold تنظیم می‌شود (شکل ۲۸).

```
Control panel - > administrative tools -> local security policy -> security settings -> account policies
```



شکل ۲۸: غیرفعال‌سازی نام کاربری پس از سه بار تلاش ناموفق برای ورود

حال در صورتی که سه مرتبه با نام کاربری و گذرواژه‌ی اشتباه تلاش برای ورود انجام شود، پیغام موجود در شکل ۲۹ ظاهر می‌شود.



شکل ۲۹: غیرفعال‌سازی نام کاربری به صورت خودکار پس از سه مرتبه تلاش ناموفق برای ورود

حال می‌توان نام کاربری را با دستور زیر از حالت قفل خارج کرد.

```
ALTER LOGIN test_user WITH PASSWORD = 'password' UNLOCK ;
```

همچنین به منظور خارج کردن نام کاربری از حالت قفل بدون اینکه نیاز به تغییر کلمه‌ی عبور باشد، دو دستور زیر اجرا می‌شوند.

```
ALTER LOGIN test_user WITH CHECK_POLICY = OFF;

ALTER LOGIN test_user WITH CHECK_POLICY = ON;
```

جدول زیر، گام‌های مربوط به ترمیم را برای رویداد ورود غیرمجاز به پایگاه‌داده نشان می‌دهد.

غیرفعال کردن/فعال‌سازی حساب کاربری به صورت دستی		
۱	غیرفعال کردن حساب کاربری	ALTER LOGIN <USER_NAME> DISABLE
۲	فعال کردن حساب کاربری	ALTER LOGIN <USER_NAME> ENABLE
قفل کردن خودکار نام کاربری پس از تعدادی تلاش ناموفق برای ورود		
۱	تنظیمات روبرو در سیستم عال ویندوز اعمال می‌شوند	Control panel - > administrative tools -> local security policy -> security settings -> account policies -> Account lockout policy -> Account lockout threshold
۲	خارج کردن حساب کاربری از حالت قفل (نیاز به تغییر رمز عبور)	ALTER LOGIN test_user WITH PASSWORD = 'password' UNLOCK ;
۳	خارج کردن حساب کاربری از حالت قفل (بدون نیاز به تغییر رمز عبور)	ALTER LOGIN test_user WITH CHECK_POLICY = OFF; ALTER LOGIN test_user WITH CHECK_POLICY = ON

۶-۵ ارائه‌ی مستندات

برای این منظور می‌بایست به هنگام بررسی جرم، جدول زیر به طور دقیق تکمیل و به مدیریت سازمان برای پیگیری‌های لازم و طرح دعوی ارائه گردد.

جدول ۵: تهیه‌ی مستند از فرآیند جرم‌شناسی برای رویداد تلاش برای یافتن نام کاربری یا کلمه‌ی عبور

تلاش برای یافتن نام کاربری یا کلمه‌ی عبور			
وقوع جرم	عدم وقوع ☐	وقوع خصمانه ☐	وقوع ناخواسته ☐
شیوه ممیزی	مشخصه‌ی ممیزی سرور ☐		
منابع رویدادنگاری	فاقد منبع رویدادنگاری است.		
شیوه یا ابزار تحلیل	فاقد شیوه یا ابزار تحلیل است.		
امکان ترمیم	قفل کردن نام کاربری دستی یا به صورت خودکار ☐		
توضیحات			

۶-۶ جمع‌بندی

در این فصل، تلاش برای ورود به پایگاه‌داده در صورت نداشتن نام کاربری یا کلمه‌ی عبور مورد بحث و بررسی قرار گرفت. برای این منظور، پس از شناسایی جرم، رویکردی به نام مشخصه‌ی ممیزی سرور برای جمع‌آوری شواهد و اطلاعات مربوط به جرم معرفی گردید. در پایان نیز برای رویکرد معرفی شده، تنظیمات مربوط به فعال‌سازی، استخراج اطلاعات و شواهد، تحلیل شواهد، ترمیم و تهیه مستندات تشریح گردید.

۷ خلاصه مطالب

جرم‌شناسی پایگاه‌داده فرآیندی است که طی آن تلاش می‌شود تا اطلاعاتی چون زمان/ چگونگی/ چرایی و فرد مجرم برای یک رخداد غیرمجاز در سامانه مشخص شود. جرم‌شناسی زمانی رخ می‌دهد که از مأمور ممیزی، کشف چگونگی وقوع نقض امنیتی و شخص مجرم درخواست شود. جرم‌شناسی پایگاه‌داده، چالش‌ها و مسائل زیادی به همراه دارد که آن را تبدیل به یک موضوع پیچیده کرده است.

با استفاده از مدل‌های فرآیند جرم‌شناسی پایگاه‌داده می‌توان به صورت ساختاریافته عملیات مربوط به جرم‌شناسی پایگاه‌داده را پیش برد. در مراحل مختلف فرآیند جرم‌شناسی، تمرکز اصلی بر روی پایگاه‌داده و اطلاعات موجود در آن برای شناسایی جرم است. در حقیقت تنها از اطلاعات ثبت‌شده در پایگاه‌های داده به منظور شناسایی جرم استفاده می‌شود و استفاده از اطلاعات ثبت‌شده بر روی سیستم عامل، داده‌های موجود

در حافظه و شبکه خارج از حوزه‌ی مورد بحث است. در یک دسته‌بندی کلی، جرم‌شناسی پایگاه‌داده شامل گام‌های زیر است:

۱. شناسایی جرم،
۲. جمع‌آوری اطلاعات و شواهد،
۳. تجزیه و تحلیل،
۴. ترمیم،
۵. ارائه‌ی مستندات.

هر سمپاد، شواهد مربوط به رویدادهای مختلف را در فایل‌های مختلف برای استفاده در تجزیه و تحلیل جرم‌شناسی ذخیره می‌کند. این به معنای آن است که برای تجزیه و تحلیل جرم‌شناسی باید نسبت به چگونگی عملکرد پایگاه‌داده و محل فایل‌ها و مصنوعات مختلف اطلاع داشت. لازم به ذکر است که جمع‌آوری مصنوعات و شواهد می‌تواند سبب تغییر در پایگاه‌داده شود، بنابراین پیش از استخراج اطلاعات از پایگاه‌داده یا خارج از پایگاه‌داده باید نسبت به این موضوع و پایدار یا ناپایدار بودن اطلاعات آگاهی پیدا کرد.

برای هر یک از رویدادهای غیرمجاز در سامانه پایگاه‌داده به هنگام بررسی جرم، جداولی برای ارائه مستندات مورد نیاز در نظر گرفته شده است که می‌بایست در طول فرآیند جرم‌شناسی به طور دقیق تکمیل و به مدیریت سازمان برای پیگیری‌های لازم و طرح دعوی ارائه گردد. جدول زیر، اطلاعات مربوط به تمامی رویدادهای غیرمجاز تشریح شده در مستند حاضر را به تصویر کشیده است.

جدول ۶: تهیه‌ی مستند از فرآیند جرم‌شناسی در پایگاه‌داده‌ی MsSQL

درج غیرمجاز در جدول			
وقوع جرم	عدم وقوع ☐	وقوع خصمانه ☐	وقوع ناخواسته ☐
شیوه ممیزی	مشخصه‌ی ممیزی پایگاه‌داده ☐		
منابع رویدادنگاری	رویدادنگاری در فایل ثبت تراکنش ☐		
شیوه یا ابزار تحلیل	فاقد شیوه یا ابزار تحلیل است.		
امکان ترمیم	حذف سطر(های) درج شده ☐		
توضیحات			
حذف غیرمجاز محتوای جدول			
وقوع جرم	عدم وقوع ☐	وقوع خصمانه ☐	وقوع ناخواسته ☐

شیوه ممیزی	مشخصه‌ی ممیزی پایگاه‌داده ☐		
منابع رویدادنگاری	رویدادنگاری در فایل ثبت تراکنش ☐		
شیوه یا ابزار تحلیل	فاقد شیوه یا ابزار تحلیل است.		
امکان ترمیم	استفاده از رویه‌های تعریف شده ☐		
توضیحات			
مشاهده غیرمجاز محتوای جدول			
وقوع جرم	عدم وقوع ☐	وقوع خصمانه ☐	وقوع ناخواسته ☐
شیوه ممیزی	مشخصه‌ی ممیزی پایگاه‌داده ☐		
منابع رویدادنگاری	رویدادنگاری در فایل ثبت تراکنش ☐		
شیوه یا ابزار تحلیل	فاقد شیوه یا ابزار تحلیل است.		
امکان ترمیم	فاقد امکان ترمیم است.		
توضیحات			
بروزرسانی غیرمجاز محتوای جدول			
وقوع جرم	عدم وقوع ☐	وقوع خصمانه ☐	وقوع ناخواسته ☐
شیوه ممیزی	مشخصه‌ی ممیزی پایگاه‌داده ☐		
منابع رویدادنگاری	رویدادنگاری در فایل ثبت تراکنش ☐		
شیوه یا ابزار تحلیل	فاقد شیوه یا ابزار تحلیل است.		
امکان ترمیم	استفاده از رویه‌های تعریف شده ☐		
توضیحات			
تغییر غیرمجاز شمای پایگاه‌داده			
وقوع جرم	عدم وقوع ☐	وقوع خصمانه ☐	وقوع ناخواسته ☐
شیوه ممیزی	مشخصه‌ی ممیزی پایگاه‌داده ☐		
منابع رویدادنگاری	رویدادنگاری در فایل ثبت تراکنش ☐		

شیوه یا ابزار تحلیل	فاقد شیوه یا ابزار تحلیل است.		
امکان ترمیم	استفاده از ApexSQL Recover ☐		
توضیحات			
تلاش برای یافتن نام کاربری یا کلمه‌ی عبور			
وقوع جرم	عدم وقوع ☐	وقوع خصمانه ☐	وقوع ناخواسته ☐
شیوه ممیزی	مشخصه‌ی ممیزی سرور ☐		
منابع رویدادنگاری	فاقد منبع رویدادنگاری است.		
شیوه یا ابزار تحلیل	فاقد شیوه یا ابزار تحلیل است.		
امکان ترمیم	قفل کردن نام کاربری دستی یا به صورت خودکار ☐		
توضیحات			

۸ منابع

[1]. DB audit and security 360, version 5.0, SoftTree Technologies, Inc.

[2]. http://www.dba-oracle.com

[3]. Al-Dhaqm, A., Razak, S.A., Othman, S.H., Nagdi, A. and Ali, A., 2016. A GENERIC DATABASE FORENSIC INVESTIGATION PROCESS MODEL. Jurnal Teknologi, 78.

[4]. R. Ramakrishnan and J. Gehrke. Database Management Systems (Third Edition). McGraw-Hill, Inc. New York, NY, USA, 2003.

[5]. G. Palmer, A Road Map for Digital Forensic Research, DFRWS Technical Report, DTR-T001-01 Final, Air Force Research Laboratory, Rome, New York, 2001.

[6]. https://docs.oracle.com

[7]. https://en.wikipedia.org/wiki/Transaction_log

[8]. J. Shital, Forensic Investigation for Database Tampering using Audit Logs, International Journal of Engineering Research & Technology (IJERT), Vol. 4 Issue 03, March 2015

[9]. K. Fowler, SQL Server database forensics, presented at the Black Hat USA Conference, 2007.

[10]. Fasan, O.M. and Olivier, M.S., 2012. On Dimensions of Reconstruction in Database Forensics. In WDFIA (pp. 97-106).

[11]. Al-Dhaqm, A., Razak, S.A., Othman, S.H., Nagdi, A. and Ali, A., 2016. A GENERIC DATABASE FORENSIC INVESTIGATION PROCESS MODEL. Jurnal Teknologi, 78.

[12]. https://solutioncenter.apexsql.com/recover-sql-server-database-using-only-a-transaction-log-file-ldf-and-old-backup-files/

[13]. H. Q. Beyers, "Database forensics: Investigating compromised database management systems", 2013.

[14]. Khanuja, H.K., Adane, D.S.: A Framework For Database Forensic Analysis. Published in Computer Science & Engineering: An International Journal (CSEIJ) 2(3) (2012)

[15]. Finnigan, P., *Oracle Incident Response and Forensics:* Preparing for and Responding to Data Breaches, 2018, Apress, Berkeley, CA.

[16]. https://dbatricksworld.com/ora-38707-media-recovery-is-not-enabled/

[17]. http://www.innovateus.net/science/what-forensics

[18]. R. Urbano, 2017, Oracle Database Administrator's Guide, 12c Release 2 (12.2)

[19]. https://www.red-gate.com/simple-talk/sql/learn-sql-server/managing-transaction-logs-in-sql-server/

[20]. https://docs.microsoft.com/en-us/sql/relational-databases/logs/the-transaction-log-sql-server

[21]. https://raresql.com/2011/10/22/how-to-recover-deleted-data-from-sql-sever/

[22]. https://raresql.com/2012/02/01/how-to-recover-modified-records-from-sql-server-part-1/

[23]. https://raresql.com/2014/02/26/sql-server-who-dropped-what-object-at-what-time/

۹ پیوست

۹-۱ پیوست الف

```
-- Script Name: Recover_Deleted_Data_Proc
-- Script Type : Recovery Procedure
-- Develop By: Muhammad Imran
-- Date Created: 15 Oct 2011
-- Modify Date: 22 Aug 2012
-- Version     : 3.1
-- Notes : Included BLOB data types for recovery.& Compatibile with
Default , CS collation , Arabic_CI_AS.

-- DROP PROCEDURE Recover_Deleted_Data_Proc

Create PROCEDURE Recover_Deleted_Data_Proc
@Database_Name NVARCHAR(MAX),
@SchemaName_n_TableName NVARCHAR(Max),
@Date_From DATETIME='1900/01/01',
@Date_To DATETIME ='9999/12/31'
AS

DECLARE @RowLogContents VARBINARY(8000)
DECLARE @TransactionID NVARCHAR(Max)
DECLARE @AllocUnitID BIGINT
```

```
DECLARE @AllocUnitName NVARCHAR(Max)
DECLARE @SQL NVARCHAR(Max)
DECLARE @Compatibility_Level INT


SELECT @Compatibility_Level=dtb.compatibility_level
FROM
master.sys.databases AS dtb WHERE dtb.name=@Database_Name

IF ISNULL(@Compatibility_Level,0)<=80
BEGIN
    RAISERROR('The compatibility level should be equal to or greater SQL
SERVER 2005 (90)',16,1)
    RETURN
END
 IF (SELECT COUNT(*) FROM INFORMATION_SCHEMA.TABLES WHERE
[TABLE_SCHEMA]+'.'+[TABLE_NAME]=@SchemaName_n_TableName)=0
BEGIN
    RAISERROR('Could not found the table in the defined database',16,1)
    RETURN
END

DECLARE @bitTable TABLE
(
  [ID] INT,
  [Bitvalue] INT
)
--Create table to set the bit position of one byte.

INSERT INTO @bitTable
SELECT 0,2 UNION ALL
SELECT 1,2 UNION ALL
SELECT 2,4 UNION ALL
SELECT 3,8 UNION ALL
SELECT 4,16 UNION ALL
SELECT 5,32 UNION ALL
SELECT 6,64 UNION ALL
SELECT 7,128

--Create table to collect the row data.
DECLARE @DeletedRecords TABLE
(
    [Row ID]             INT IDENTITY(1,1),
    [RowLogContents]     VARBINARY(8000),
    [AllocUnitID]        BIGINT,
    [Transaction ID]     NVARCHAR(Max),
    [FixedLengthData]    SMALLINT,
    [TotalNoOfCols]      SMALLINT,
    [NullBitMapLength]   SMALLINT,
    [NullBytes]          VARBINARY(8000),
    [TotalNoofVarCols]   SMALLINT,
    [ColumnOffsetArray] VARBINARY(8000),
    [VarColumnStart]     SMALLINT,
    [Slot ID]            INT,
    [NullBitMap]         VARCHAR(MAX)

)
```

```
--Create a common table expression to get all the row data plus how many
bytes we have for each row.
;WITH RowData AS (
SELECT

[RowLog Contents 0] AS [RowLogContents]

,[AllocUnitID] AS [AllocUnitID]

,[Transaction ID] AS [Transaction ID]

--[Fixed Length Data] = Substring (RowLog content 0, Status Bit A+ Status
Bit B + 1,2 bytes)
,CONVERT(SMALLINT, CONVERT(BINARY(2), REVERSE(SUBSTRING([RowLog Contents
0], 2 + 1, 2)))) AS [FixedLengthData]  --@FixedLengthData

-- [TotalnoOfCols] =  Substring (RowLog content 0, [Fixed Length Data] +
1,2 bytes)
,CONVERT(INT, CONVERT(BINARY(2), REVERSE(SUBSTRING([RowLog Contents 0],
CONVERT(SMALLINT, CONVERT(BINARY(2)
,REVERSE(SUBSTRING([RowLog Contents 0], 2 + 1, 2)))) + 1, 2)))) as
[TotalNoOfCols]

--[NullBitMapLength]=ceiling([Total No of Columns] /8.0)
,CONVERT(INT, ceiling(CONVERT(INT, CONVERT(BINARY(2),
REVERSE(SUBSTRING([RowLog Contents 0], CONVERT(SMALLINT,
CONVERT(BINARY(2)
,REVERSE(SUBSTRING([RowLog Contents 0], 2 + 1, 2)))) + 1, 2))))/8.0)) as
[NullBitMapLength]

--[Null Bytes] = Substring (RowLog content 0, Status Bit A+ Status Bit B
+ [Fixed Length Data] +1, [NullBitMapLength] )
,SUBSTRING([RowLog Contents 0], CONVERT(SMALLINT, CONVERT(BINARY(2),
REVERSE(SUBSTRING([RowLog Contents 0], 2 + 1, 2)))) + 3,
CONVERT(INT, ceiling(CONVERT(INT, CONVERT(BINARY(2),
REVERSE(SUBSTRING([RowLog Contents 0], CONVERT(SMALLINT,
CONVERT(BINARY(2)
,REVERSE(SUBSTRING([RowLog Contents 0], 2 + 1, 2)))) + 1, 2))))/8.0))) as
[NullBytes]

--[TotalNoofVarCols] = Substring (RowLog content 0, Status Bit A+ Status
Bit B + [Fixed Length Data] +1, [Null Bitmap length] + 2 )
,(CASE WHEN SUBSTRING([RowLog Contents 0], 1, 1) In (0x10,0x30,0x70) THEN
CONVERT(INT, CONVERT(BINARY(2), REVERSE(SUBSTRING([RowLog Contents 0],
CONVERT(SMALLINT, CONVERT(BINARY(2), REVERSE(SUBSTRING([RowLog Contents
0], 2 + 1, 2)))) + 3
+ CONVERT(INT, ceiling(CONVERT(INT, CONVERT(BINARY(2),
REVERSE(SUBSTRING([RowLog Contents 0], CONVERT(SMALLINT,
CONVERT(BINARY(2)
,REVERSE(SUBSTRING([RowLog Contents 0], 2 + 1, 2)))) + 1, 2))))/8.0)),
2))))  ELSE null  END) AS [TotalNoofVarCols]

--[ColumnOffsetArray]= Substring (RowLog content 0, Status Bit A+ Status
Bit B + [Fixed Length Data] +1, [Null Bitmap length] + 2 ,
[TotalNoofVarCols]*2 )
,(CASE WHEN SUBSTRING([RowLog Contents 0], 1, 1) In (0x10,0x30,0x70) THEN
SUBSTRING([RowLog Contents 0]
```

```
, CONVERT(SMALLINT, CONVERT(BINARY(2), REVERSE(SUBSTRING([RowLog Contents
0], 2 + 1, 2)))) + 3
+ CONVERT(INT, ceiling(CONVERT(INT, CONVERT(BINARY(2),
REVERSE(SUBSTRING([RowLog Contents 0], CONVERT(SMALLINT,
CONVERT(BINARY(2)
,REVERSE(SUBSTRING([RowLog Contents 0], 2 + 1, 2)))) + 1, 2))))/8.0)) + 2
, (CASE WHEN SUBSTRING([RowLog Contents 0], 1, 1) In (0x10,0x30,0x70)
THEN
CONVERT(INT, CONVERT(BINARY(2), REVERSE(SUBSTRING([RowLog Contents 0],
CONVERT(SMALLINT, CONVERT(BINARY(2), REVERSE(SUBSTRING([RowLog Contents
0], 2 + 1, 2)))) + 3
+ CONVERT(INT, ceiling(CONVERT(INT, CONVERT(BINARY(2),
REVERSE(SUBSTRING([RowLog Contents 0], CONVERT(SMALLINT,
CONVERT(BINARY(2)
,REVERSE(SUBSTRING([RowLog Contents 0], 2 + 1, 2)))) + 1, 2))))/8.0)),
2))))  ELSE null  END)
* 2)  ELSE null  END) AS [ColumnOffsetArray]

--  Variable column Start = Status Bit A+ Status Bit B + [Fixed Length
Data] + [Null Bitmap length] + 2+([TotalNoofVarCols]*2)
,CASE WHEN SUBSTRING([RowLog Contents 0], 1, 1)In (0x10,0x30,0x70)
THEN  (
CONVERT(SMALLINT, CONVERT(BINARY(2), REVERSE(SUBSTRING([RowLog Contents
0], 2 + 1, 2)))) + 4

+ CONVERT(INT, ceiling(CONVERT(INT, CONVERT(BINARY(2),
REVERSE(SUBSTRING([RowLog Contents 0], CONVERT(SMALLINT,
CONVERT(BINARY(2)
,REVERSE(SUBSTRING([RowLog Contents 0], 2 + 1, 2)))) + 1, 2))))/8.0))

+ ((CASE WHEN SUBSTRING([RowLog Contents 0], 1, 1) In (0x10,0x30,0x70)
THEN
CONVERT(INT, CONVERT(BINARY(2), REVERSE(SUBSTRING([RowLog Contents 0],
CONVERT(SMALLINT, CONVERT(BINARY(2), REVERSE(SUBSTRING([RowLog Contents
0], 2 + 1, 2)))) + 3
+ CONVERT(INT, ceiling(CONVERT(INT, CONVERT(BINARY(2),
REVERSE(SUBSTRING([RowLog Contents 0], CONVERT(SMALLINT,
CONVERT(BINARY(2)
,REVERSE(SUBSTRING([RowLog Contents 0], 2 + 1, 2)))) + 1, 2))))/8.0)),
2))))  ELSE null  END) * 2))

ELSE null End AS [VarColumnStart]
,[Slot ID]
FROM sys.fn_dblog(NULL, NULL)
WHERE
AllocUnitId IN
(SELECT [Allocation_unit_id] FROM sys.allocation_units allocunits
INNER JOIN sys.partitions partitions ON (allocunits.type IN (1, 3)
AND partitions.hobt_id = allocunits.container_id) OR (allocunits.type = 2
AND partitions.partition_id = allocunits.container_id)
WHERE object_id=object_ID('' + @SchemaName_n_TableName + ''))

AND Context IN ('LCX_MARK_AS_GHOST', 'LCX_HEAP') AND Operation in
('LOP_DELETE_ROWS')
And SUBSTRING([RowLog Contents 0], 1, 1)In (0x10,0x30,0x70)

/*Use this subquery to filter the date*/
```

```
AND [TRANSACTION ID] IN (SELECT DISTINCT [TRANSACTION ID] FROM
sys.fn_dblog(NULL, NULL)
WHERE Context IN ('LCX_NULL') AND Operation in ('LOP_BEGIN_XACT')
And [Transaction Name] In ('DELETE','user_transaction')
And  CONVERT(NVARCHAR(11),[Begin Time]) BETWEEN @Date_From AND
@Date_To)),

--Use this technique to repeate the row till the no of bytes of the row.
N1 (n) AS (SELECT 1 UNION ALL SELECT 1),
N2 (n) AS (SELECT 1 FROM N1 AS X, N1 AS Y),
N3 (n) AS (SELECT 1 FROM N2 AS X, N2 AS Y),
N4 (n) AS (SELECT ROW_NUMBER() OVER(ORDER BY X.n)
           FROM N3 AS X, N3 AS Y)



INSERT INTO @DeletedRecords
SELECT  RowLogContents
        ,[AllocUnitID]
        ,[Transaction ID]
        ,[FixedLengthData]
        ,[TotalNoOfCols]
        ,[NullBitMapLength]
        ,[NullBytes]
        ,[TotalNoofVarCols]
        ,[ColumnOffsetArray]
        ,[VarColumnStart]
        ,[Slot ID]
         ---Get the Null value against each column (1 means null zero
means not null)
        ,[NullBitMap]=(REPLACE(STUFF((SELECT ',' +
        (CASE WHEN [ID]=0 THEN CONVERT(NVARCHAR(1),(SUBSTRING(NullBytes,
n, 1) % 2))  ELSE CONVERT(NVARCHAR(1),((SUBSTRING(NullBytes, n, 1) /
[Bitvalue]) % 2)) END) --as [nullBitMap]

FROM
N4 AS Nums
Join RowData AS C ON n<=NullBitMapLength
Cross Join @bitTable WHERE C.[RowLogContents]=D.[RowLogContents] ORDER BY
[RowLogContents],n ASC FOR XML PATH('')),1,1,''),',','''))
FROM RowData D

IF (SELECT COUNT(*) FROM @DeletedRecords)=0
BEGIN
    RAISERROR('There is no data in the log as per the search
criteria',16,1)
    RETURN
END

DECLARE @ColumnNameAndData TABLE
(
 [Row ID]            int,
 [Rowlogcontents]    varbinary(Max),
 [NAME]              sysname,
 [nullbit]           smallint,
 [leaf_offset]       smallint,
 [length]            smallint,
```

```
 [system_type_id]    tinyint,
 [bitpos]            tinyint,
 [xprec]             tinyint,
 [xscale]            tinyint,
 [is_null]           int,
 [Column value Size]int,
 [Column Length]     int,
 [hex_Value]         varbinary(max),
 [Slot ID]           int,
 [Update]            int
)

--Create common table expression and join it with the rowdata table
-- to get each column details
/*This part is for variable data columns*/
--@RowLogContents,
--(col.columnOffValue - col.columnLength) + 1,
--col.columnLength
--)
INSERT INTO @ColumnNameAndData
SELECT
[Row ID],
Rowlogcontents,
NAME ,
cols.leaf_null_bit AS nullbit,
leaf_offset,
ISNULL(syscolumns.length, cols.max_length) AS [length],
cols.system_type_id,
cols.leaf_bit_position AS bitpos,
ISNULL(syscolumns.xprec, cols.precision) AS xprec,
ISNULL(syscolumns.xscale, cols.scale) AS xscale,
SUBSTRING([nullBitMap], cols.leaf_null_bit, 1) AS is_null,
(CASE WHEN leaf_offset<1 and SUBSTRING([nullBitMap], cols.leaf_null_bit,
1)=0
THEN
(Case When CONVERT(INT, CONVERT(BINARY(2), REVERSE (SUBSTRING
([ColumnOffsetArray], (2 * leaf_offset*-1) - 1, 2)))) >30000
THEN
CONVERT(INT, CONVERT(BINARY(2), REVERSE (SUBSTRING ([ColumnOffsetArray],
(2 * leaf_offset*-1) - 1, 2)))) - POWER(2, 15)
ELSE
CONVERT(INT, CONVERT(BINARY(2), REVERSE (SUBSTRING ([ColumnOffsetArray],
(2 * leaf_offset*-1) - 1, 2))))
END)
END)  AS [Column value Size],

(CASE WHEN leaf_offset<1 and SUBSTRING([nullBitMap], cols.leaf_null_bit,
1)=0  THEN
(Case

When CONVERT(INT, CONVERT(BINARY(2), REVERSE (SUBSTRING
([ColumnOffsetArray], (2 * leaf_offset*-1) - 1, 2)))) >30000 And
ISNULL(NULLIF(CONVERT(INT, CONVERT(BINARY(2), REVERSE (SUBSTRING
([ColumnOffsetArray], (2 * ((leaf_offset*-1) - 1)) - 1, 2)))), 0),
[varColumnStart])<30000
THEN  (Case When [System_type_id]In (35,34,99) Then 16 else 24  end)
```

```
When CONVERT(INT, CONVERT(BINARY(2), REVERSE (SUBSTRING
([ColumnOffsetArray], (2 * leaf_offset*-1) - 1, 2)))) >30000 And
ISNULL(NULLIF(CONVERT(INT, CONVERT(BINARY(2), REVERSE (SUBSTRING
([ColumnOffsetArray], (2 * ((leaf_offset*-1) - 1)) - 1, 2)))), 0),
[varColumnStart])>30000
THEN  (Case When [System_type_id]In (35,34,99) Then 16 else 24  end) --24

When CONVERT(INT, CONVERT(BINARY(2), REVERSE (SUBSTRING
([ColumnOffsetArray], (2 * leaf_offset*-1) - 1, 2)))) <30000 And
ISNULL(NULLIF(CONVERT(INT, CONVERT(BINARY(2), REVERSE (SUBSTRING
([ColumnOffsetArray], (2 * ((leaf_offset*-1) - 1)) - 1, 2)))), 0),
[varColumnStart])<30000
THEN (CONVERT(INT, CONVERT(BINARY(2), REVERSE (SUBSTRING
([ColumnOffsetArray], (2 * leaf_offset*-1) - 1, 2))))
- ISNULL(NULLIF(CONVERT(INT, CONVERT(BINARY(2), REVERSE (SUBSTRING
([ColumnOffsetArray], (2 * ((leaf_offset*-1) - 1)) - 1, 2)))), 0),
[varColumnStart]))

When CONVERT(INT, CONVERT(BINARY(2), REVERSE (SUBSTRING
([ColumnOffsetArray], (2 * leaf_offset*-1) - 1, 2)))) <30000 And
ISNULL(NULLIF(CONVERT(INT, CONVERT(BINARY(2), REVERSE (SUBSTRING
([ColumnOffsetArray], (2 * ((leaf_offset*-1) - 1)) - 1, 2)))), 0),
[varColumnStart])>30000

THEN POWER(2, 15) +CONVERT(INT, CONVERT(BINARY(2), REVERSE (SUBSTRING
([ColumnOffsetArray], (2 * leaf_offset*-1) - 1, 2))))
- ISNULL(NULLIF(CONVERT(INT, CONVERT(BINARY(2), REVERSE (SUBSTRING
([ColumnOffsetArray], (2 * ((leaf_offset*-1) - 1)) - 1, 2)))), 0),
[varColumnStart])

END)

END) AS [Column Length]

,(CASE WHEN SUBSTRING([nullBitMap], cols.leaf_null_bit, 1)=1 THEN  NULL
ELSE
 SUBSTRING
 (
 Rowlogcontents,
 (

(Case When CONVERT(INT, CONVERT(BINARY(2), REVERSE (SUBSTRING
([ColumnOffsetArray], (2 * leaf_offset*-1) - 1, 2)))) >30000
THEN
CONVERT(INT, CONVERT(BINARY(2), REVERSE (SUBSTRING ([ColumnOffsetArray],
(2 * leaf_offset*-1) - 1, 2)))) - POWER(2, 15)
ELSE
CONVERT(INT, CONVERT(BINARY(2), REVERSE (SUBSTRING ([ColumnOffsetArray],
(2 * leaf_offset*-1) - 1, 2))))
END)

 -
(Case When CONVERT(INT, CONVERT(BINARY(2), REVERSE (SUBSTRING
([ColumnOffsetArray], (2 * leaf_offset*-1) - 1, 2)))) >30000 And
ISNULL(NULLIF(CONVERT(INT, CONVERT(BINARY(2), REVERSE (SUBSTRING
([ColumnOffsetArray], (2 * ((leaf_offset*-1) - 1)) - 1, 2)))), 0),
[varColumnStart])<30000
```

```
THEN  (Case When [System_type_id]In (35,34,99) Then 16 else 24  end) --24
When CONVERT(INT, CONVERT(BINARY(2), REVERSE (SUBSTRING
([ColumnOffsetArray], (2 * leaf_offset*-1) - 1, 2)))) >30000 And
ISNULL(NULLIF(CONVERT(INT, CONVERT(BINARY(2), REVERSE (SUBSTRING
([ColumnOffsetArray], (2 * ((leaf_offset*-1) - 1)) - 1, 2)))), 0),
[varColumnStart])>30000

THEN  (Case When [System_type_id]In (35,34,99) Then 16 else 24  end) --24
When CONVERT(INT, CONVERT(BINARY(2), REVERSE (SUBSTRING
([ColumnOffsetArray], (2 * leaf_offset*-1) - 1, 2)))) <30000 And
ISNULL(NULLIF(CONVERT(INT, CONVERT(BINARY(2), REVERSE (SUBSTRING
([ColumnOffsetArray], (2 * ((leaf_offset*-1) - 1)) - 1, 2)))), 0),
[varColumnStart])<30000

THEN CONVERT(INT, CONVERT(BINARY(2), REVERSE (SUBSTRING
([ColumnOffsetArray], (2 * leaf_offset*-1) - 1, 2))))
- ISNULL(NULLIF(CONVERT(INT, CONVERT(BINARY(2), REVERSE (SUBSTRING
([ColumnOffsetArray], (2 * ((leaf_offset*-1) - 1)) - 1, 2)))), 0),
[varColumnStart])

When CONVERT(INT, CONVERT(BINARY(2), REVERSE (SUBSTRING
([ColumnOffsetArray], (2 * leaf_offset*-1) - 1, 2)))) <30000 And
ISNULL(NULLIF(CONVERT(INT, CONVERT(BINARY(2), REVERSE (SUBSTRING
([ColumnOffsetArray], (2 * ((leaf_offset*-1) - 1)) - 1, 2)))), 0),
[varColumnStart])>30000

THEN POWER(2, 15) +CONVERT(INT, CONVERT(BINARY(2), REVERSE (SUBSTRING
([ColumnOffsetArray], (2 * leaf_offset*-1) - 1, 2))))
- ISNULL(NULLIF(CONVERT(INT, CONVERT(BINARY(2), REVERSE (SUBSTRING
([ColumnOffsetArray], (2 * ((leaf_offset*-1) - 1)) - 1, 2)))), 0),
[varColumnStart])

END)

) + 1,
(Case When CONVERT(INT, CONVERT(BINARY(2), REVERSE (SUBSTRING
([ColumnOffsetArray], (2 * leaf_offset*-1) - 1, 2)))) >30000 And
ISNULL(NULLIF(CONVERT(INT, CONVERT(BINARY(2), REVERSE (SUBSTRING
([ColumnOffsetArray], (2 * ((leaf_offset*-1) - 1)) - 1, 2)))), 0),
[varColumnStart])<30000

THEN  (Case When [System_type_id] In (35,34,99) Then 16 else 24  end) --
24
When CONVERT(INT, CONVERT(BINARY(2), REVERSE (SUBSTRING
([ColumnOffsetArray], (2 * leaf_offset*-1) - 1, 2)))) >30000 And
ISNULL(NULLIF(CONVERT(INT, CONVERT(BINARY(2), REVERSE (SUBSTRING
([ColumnOffsetArray], (2 * ((leaf_offset*-1) - 1)) - 1, 2)))), 0),
[varColumnStart])>30000

THEN  (Case When [System_type_id] In (35,34,99) Then 16 else 24  end) --
24
When CONVERT(INT, CONVERT(BINARY(2), REVERSE (SUBSTRING
([ColumnOffsetArray], (2 * leaf_offset*-1) - 1, 2)))) <30000 And
ISNULL(NULLIF(CONVERT(INT, CONVERT(BINARY(2), REVERSE (SUBSTRING
([ColumnOffsetArray], (2 * ((leaf_offset*-1) - 1)) - 1, 2)))), 0),
[varColumnStart])<30000
```

```
THEN ABS(CONVERT(INT, CONVERT(BINARY(2), REVERSE (SUBSTRING
([ColumnOffsetArray], (2 * leaf_offset*-1) - 1, 2))))
- ISNULL(NULLIF(CONVERT(INT, CONVERT(BINARY(2), REVERSE (SUBSTRING
([ColumnOffsetArray], (2 * ((leaf_offset*-1) - 1)) - 1, 2)))), 0),
[varColumnStart]))

When CONVERT(INT, CONVERT(BINARY(2), REVERSE (SUBSTRING
([ColumnOffsetArray], (2 * leaf_offset*-1) - 1, 2)))) <30000 And
ISNULL(NULLIF(CONVERT(INT, CONVERT(BINARY(2), REVERSE (SUBSTRING
([ColumnOffsetArray], (2 * ((leaf_offset*-1) - 1)) - 1, 2)))), 0),
[varColumnStart])>30000

THEN POWER(2, 15) +CONVERT(INT, CONVERT(BINARY(2), REVERSE (SUBSTRING
([ColumnOffsetArray], (2 * leaf_offset*-1) - 1, 2))))
- ISNULL(NULLIF(CONVERT(INT, CONVERT(BINARY(2), REVERSE (SUBSTRING
([ColumnOffsetArray], (2 * ((leaf_offset*-1) - 1)) - 1, 2)))), 0),
[varColumnStart])

END)
)

END) AS hex_Value
,[Slot ID]
,0
FROM @DeletedRecords A
Inner Join sys.allocation_units allocunits On
A.[AllocUnitId]=allocunits.[Allocation_Unit_Id]
INNER JOIN sys.partitions partitions ON (allocunits.type IN (1, 3)
AND partitions.hobt_id = allocunits.container_id) OR (allocunits.type = 2
AND partitions.partition_id = allocunits.container_id)
INNER JOIN sys.system_internals_partition_columns cols ON
cols.partition_id = partitions.partition_id
LEFT OUTER JOIN syscolumns ON syscolumns.id = partitions.object_id AND
syscolumns.colid = cols.partition_column_id
WHERE leaf_offset<0
UNION
/*This part is for fixed data columns*/
SELECT
[Row ID],
Rowlogcontents,
NAME ,
cols.leaf_null_bit AS nullbit,
leaf_offset,
ISNULL(syscolumns.length, cols.max_length) AS [length],
cols.system_type_id,
cols.leaf_bit_position AS bitpos,
ISNULL(syscolumns.xprec, cols.precision) AS xprec,
ISNULL(syscolumns.xscale, cols.scale) AS xscale,
SUBSTRING([nullBitMap], cols.leaf_null_bit, 1) AS is_null,
(SELECT TOP 1 ISNULL(SUM(CASE WHEN C.leaf_offset >1 THEN max_length ELSE
0 END),0) FROM
sys.system_internals_partition_columns C WHERE cols.partition_id
=C.partition_id And C.leaf_null_bit<cols.leaf_null_bit)+5 AS [Column
value Size],
syscolumns.length AS [Column Length]
```

```
,CASE WHEN SUBSTRING([nullBitMap], cols.leaf_null_bit, 1)=1 THEN NULL
ELSE
SUBSTRING
(
Rowlogcontents,(SELECT TOP 1 ISNULL(SUM(CASE WHEN C.leaf_offset >1 And
C.leaf_bit_position=0 THEN max_length ELSE 0 END),0) FROM
sys.system_internals_partition_columns C where cols.partition_id
=C.partition_id And C.leaf_null_bit<cols.leaf_null_bit)+5
,syscolumns.length) END AS hex_Value
,[Slot ID]
,0
FROM @DeletedRecords A
Inner Join sys.allocation_units allocunits ON
A.[AllocUnitId]=allocunits.[Allocation_Unit_Id]
INNER JOIN sys.partitions partitions ON (allocunits.type IN (1, 3)
AND partitions.hobt_id = allocunits.container_id) OR (allocunits.type = 2
AND partitions.partition_id = allocunits.container_id)
INNER JOIN sys.system_internals_partition_columns cols ON
cols.partition_id = partitions.partition_id
LEFT OUTER JOIN syscolumns ON syscolumns.id = partitions.object_id AND
syscolumns.colid = cols.partition_column_id
WHERE leaf_offset>0
Order By nullbit

Declare @BitColumnByte as int
Select @BitColumnByte=CONVERT(INT, ceiling( Count(*)/8.0)) from
@ColumnNameAndData Where [System_Type_id]=104

;With N1 (n) AS (SELECT 1 UNION ALL SELECT 1),
N2 (n) AS (SELECT 1 FROM N1 AS X, N1 AS Y),
N3 (n) AS (SELECT 1 FROM N2 AS X, N2 AS Y),
N4 (n) AS (SELECT ROW_NUMBER() OVER(ORDER BY X.n)
           FROM N3 AS X, N3 AS Y),
CTE As(
Select RowLogContents,[nullbit]

,[BitMap]=Convert(varbinary(1),Convert(int,Substring((REPLACE(STUFF((SELE
CT ',' +
        (CASE WHEN [ID]=0 THEN CONVERT(NVARCHAR(1),(SUBSTRING(hex_Value,
n, 1) % 2))  ELSE CONVERT(NVARCHAR(1),((SUBSTRING(hex_Value, n, 1) /
[Bitvalue]) % 2)) END) --as [nullBitMap]

from N4 AS Nums
Join @ColumnNameAndData AS C ON n<=@BitColumnByte And
[System_Type_id]=104 And bitpos=0
Cross Join @bitTable WHERE C.[RowLogContents]=D.[RowLogContents] ORDER BY
[RowLogContents],n ASC FOR XML PATH('')),1,1,''),',',''),bitpos+1,1)))
FROM @ColumnNameAndData D Where  [System_Type_id]=104)

Update A Set [hex_Value]=[BitMap]
from @ColumnNameAndData  A
Inner Join CTE B On A.[RowLogContents]=B.[RowLogContents]
And A.[nullbit]=B.[nullbit]


/**************Check for BLOB DATA TYPES*****************************/
DECLARE @Fileid INT
```

```
DECLARE @Pageid INT
DECLARE @Slotid INT
DECLARE @CurrentLSN INT
DECLARE @LinkID INT
DECLARE @Context VARCHAR(50)
DECLARE @ConsolidatedPageID VARCHAR(MAX)
DECLARE @LCX_TEXT_MIX VARBINARY(MAX)

declare @temppagedata table
(
[ParentObject] sysname,
[Object] sysname,
[Field] sysname,
[Value] sysname)

declare @pagedata table
(
[Page ID] sysname,
[File IDS] int,
[Page IDS] int,
[AllocUnitId] bigint,
[ParentObject] sysname,
[Object] sysname,
[Field] sysname,
[Value] sysname)

DECLARE @ModifiedRawData TABLE
(
  [ID] INT IDENTITY(1,1),
  [PAGE ID] VARCHAR(MAX),
  [FILE IDS] INT,
  [PAGE IDS] INT,
  [Slot ID]  INT,
  [AllocUnitId] BIGINT,
  [RowLog Contents 0_var] VARCHAR(Max),
  [RowLog Length] VARCHAR(50),
  [RowLog Len] INT,
  [RowLog Contents 0] VARBINARY(Max),
  [Link ID] INT default (0),
  [Update] INT
)

            DECLARE Page_Data_Cursor CURSOR FOR
            /*We need to filter LOP_MODIFY_ROW,LOP_MODIFY_COLUMNS from
log for deleted records of BLOB data type& Get its Slot No, Page ID &
AllocUnit ID*/
            SELECT LTRIM(RTRIM(Replace([Description],'Deallocated','')))
AS [PAGE ID]
            ,[Slot ID],[AllocUnitId],NULL AS [RowLog Contents 0],NULL AS
[RowLog Contents 0],Context
            FROM    sys.fn_dblog(NULL, NULL)
            WHERE
            AllocUnitId IN
            (SELECT [Allocation_unit_id] FROM sys.allocation_units
allocunits
            INNER JOIN sys.partitions partitions ON (allocunits.type IN
(1, 3)
```

```
            AND partitions.hobt_id = allocunits.container_id) OR
(allocunits.type = 2
            AND partitions.partition_id = allocunits.container_id)
            WHERE object_id=object_ID('' + @SchemaName_n_TableName + ''))
            AND Operation IN ('LOP_MODIFY_ROW') AND [Context] IN
('LCX_PFS')
            AND Description Like '%Deallocated%'
            /*Use this subquery to filter the date*/
            AND [TRANSACTION ID] IN (SELECT DISTINCT [TRANSACTION ID]
FROM    sys.fn_dblog(NULL, NULL)
            WHERE Context IN ('LCX_NULL') AND Operation in
('LOP_BEGIN_XACT')
            AND [Transaction Name]='DELETE'
            AND  CONVERT(NVARCHAR(11),[Begin Time]) BETWEEN @Date_From
AND @Date_To)
            GROUP BY [Description],[Slot ID],[AllocUnitId],Context

            UNION

            SELECT [PAGE ID],[Slot ID],[AllocUnitId]
            ,Substring([RowLog Contents 0],15,LEN([RowLog Contents 0]))
AS [RowLog Contents 0]
            ,CONVERT(INT,Substring([RowLog Contents 0],7,2)),Context --
,CAST(RIGHT([Current LSN],4) AS INT) AS [Current LSN]
            FROM    sys.fn_dblog(NULL, NULL)
            WHERE
             AllocUnitId IN
            (SELECT [Allocation_unit_id] FROM sys.allocation_units
allocunits
            INNER JOIN sys.partitions partitions ON (allocunits.type IN
(1, 3)
            AND partitions.hobt_id = allocunits.container_id) OR
(allocunits.type = 2
            AND partitions.partition_id = allocunits.container_id)
            WHERE object_id=object_ID('' + @SchemaName_n_TableName + ''))
            AND Context IN ('LCX_TEXT_MIX') AND Operation in
('LOP_DELETE_ROWS')
            /*Use this subquery to filter the date*/
            AND [TRANSACTION ID] IN (SELECT DISTINCT [TRANSACTION ID]
FROM    sys.fn_dblog(NULL, NULL)
            WHERE Context IN ('LCX_NULL') AND Operation in
('LOP_BEGIN_XACT')
            And [Transaction Name]='DELETE'
            And  CONVERT(NVARCHAR(11),[Begin Time]) BETWEEN @Date_From
AND @Date_To)

            /*****************************************/

        OPEN Page_Data_Cursor

        FETCH NEXT FROM Page_Data_Cursor INTO @ConsolidatedPageID,
@Slotid,@AllocUnitID,@LCX_TEXT_MIX,@LinkID,@Context

        WHILE @@FETCH_STATUS = 0
        BEGIN
            DECLARE @hex_pageid AS VARCHAR(Max)
            /*Page ID contains File Number and page number It looks like
```

```
0001:00000130.
              In this example 0001 is file Number &  00000130 is Page
Number & These numbers are in Hex format*/
            SET
@Fileid=SUBSTRING(@ConsolidatedPageID,0,CHARINDEX(':',@ConsolidatedPageID
)) -- Seperate File ID from Page ID

            SET @hex_pageid ='0x'+
SUBSTRING(@ConsolidatedPageID,CHARINDEX(':',@ConsolidatedPageID)+1,Len(@C
onsolidatedPageID))  ---Seperate the page ID
            SELECT @Pageid=Convert(INT,cast('' AS
XML).value('xs:hexBinary(substring(sql:variable("@hex_pageid"),sql:column
("t.pos")) )', 'varbinary(max)')) -- Convert Page ID from hex to integer
            FROM (SELECT CASE substring(@hex_pageid, 1, 2) WHEN '0x' THEN
3 ELSE 0 END) AS t(pos)

            IF @Context='LCX_PFS'
              BEGIN
                        DELETE @temppagedata
                        INSERT INTO @temppagedata EXEC( 'DBCC PAGE(' +
@DataBase_Name + ', ' + @fileid + ', ' + @pageid + ', 1) with
tableresults,no_infomsgs;');
                        INSERT INTO @pagedata SELECT
@ConsolidatedPageID,@fileid,@pageid,@AllocUnitID,[ParentObject],[Object],
[Field] ,[Value] FROM @temppagedata
              END
            ELSE IF @Context='LCX_TEXT_MIX'
              BEGIN
                        INSERT INTO  @ModifiedRawData SELECT
@ConsolidatedPageID,@fileid,@pageid,@Slotid,@AllocUnitID,NULL,0,CONVERT(I
NT,CONVERT(VARBINARY,REVERSE(SUBSTRING(@LCX_TEXT_MIX,11,2)))),@LCX_TEXT_M
IX,@LinkID,0
              END
            FETCH NEXT FROM Page_Data_Cursor INTO  @ConsolidatedPageID,
@Slotid,@AllocUnitID,@LCX_TEXT_MIX,@LinkID,@Context
        END

    CLOSE Page_Data_Cursor
    DEALLOCATE Page_Data_Cursor

    DECLARE @Newhexstring VARCHAR(MAX);

    --The data is in multiple rows in the page, so we need to convert it
into one row as a single hex value.
    --This hex value is in string format
    INSERT INTO @ModifiedRawData ([PAGE ID],[FILE IDS],[PAGE IDS],[Slot
ID],[AllocUnitId]
    ,[RowLog Contents 0_var]
    , [RowLog Length])
    SELECT [Page ID],[FILE IDS],[PAGE
IDS],Substring([ParentObject],CHARINDEX('Slot', [ParentObject])+4,
(CHARINDEX('Offset', [ParentObject])-(CHARINDEX('Slot',
[ParentObject])+4))-2 ) as [Slot ID]
    ,[AllocUnitId]
    ,Substring((
    SELECT
    REPLACE(STUFF((SELECT
```

```
REPLACE(SUBSTRING([Value],CHARINDEX(':',[Value])+1,CHARINDEX('†',[Value])
-CHARINDEX(':',[Value])),'†','')
    FROM @pagedata C  WHERE B.[Page ID]= C.[Page ID] And
Substring(B.[ParentObject],CHARINDEX('Slot', B.[ParentObject])+4,
(CHARINDEX('Offset', B.[ParentObject])-(CHARINDEX('Slot',
B.[ParentObject])+4)) )=Substring(C.[ParentObject],CHARINDEX('Slot',
C.[ParentObject])+4, (CHARINDEX('Offset', C.[ParentObject])-
(CHARINDEX('Slot', C.[ParentObject])+4)) ) And
    [Object] Like '%Memory Dump%'  Order By '0x'+
LEFT([Value],CHARINDEX(':',[Value])-1)
    FOR XML PATH('') ),1,1,'') ,' ','')
    ),1,20000) AS [Value]

    ,
     Substring((
    SELECT '0x' +REPLACE(STUFF((SELECT
REPLACE(SUBSTRING([Value],CHARINDEX(':',[Value])+1,CHARINDEX('†',[Value])
-CHARINDEX(':',[Value])),'†','')
    FROM @pagedata C  WHERE B.[Page ID]= C.[Page ID] And
Substring(B.[ParentObject],CHARINDEX('Slot', B.[ParentObject])+4,
(CHARINDEX('Offset', B.[ParentObject])-(CHARINDEX('Slot',
B.[ParentObject])+4)) )=Substring(C.[ParentObject],CHARINDEX('Slot',
C.[ParentObject])+4, (CHARINDEX('Offset', C.[ParentObject])-
(CHARINDEX('Slot', C.[ParentObject])+4)) ) And
    [Object] Like '%Memory Dump%'  Order By '0x'+
LEFT([Value],CHARINDEX(':',[Value])-1)
    FOR XML PATH('') ),1,1,'') ,' ','')
    ),7,4) AS [Length]

    From @pagedata B
    Where [Object] Like '%Memory Dump%'
    Group By [Page ID],[FILE IDS],[PAGE
IDS],[ParentObject],[AllocUnitId]--,[Current LSN]
    Order By [Slot ID]

    UPDATE @ModifiedRawData  SET [RowLog Len] =
CONVERT(VARBINARY(8000),REVERSE(cast('' AS
XML).value('xs:hexBinary(substring(sql:column("[RowLog Length]"),0))',
'varbinary(Max)')))
    FROM @ModifiedRawData Where [LINK ID]=0

    UPDATE @ModifiedRawData  SET [RowLog Contents 0] =cast('' AS
XML).value('xs:hexBinary(substring(sql:column("[RowLog Contents
0_var]"),0))', 'varbinary(Max)')
    FROM @ModifiedRawData Where [LINK ID]=0

    Update B Set B.[RowLog Contents 0] =
    (CASE WHEN A.[RowLog Contents 0] IS NOT NULL AND C.[RowLog Contents
0] IS NOT NULL THEN  A.[RowLog Contents 0]+C.[RowLog Contents 0]
        WHEN A.[RowLog Contents 0] IS NULL AND C.[RowLog Contents 0] IS
NOT NULL THEN  C.[RowLog Contents 0]
        WHEN A.[RowLog Contents 0] IS NOT NULL AND C.[RowLog Contents 0]
IS NULL THEN  A.[RowLog Contents 0]
        END)
    ,B.[Update]=ISNULL(B.[Update],0)+1
    from @ModifiedRawData B
    LEFT Join @ModifiedRawData A On A.[Page
```

```
IDS]=Convert(int,Convert(Varbinary(Max),Reverse(Substring(B.[RowLog
Contents 0],15+14,2))))
    And A.[File
IDS]=Convert(int,Convert(Varbinary(Max),Reverse(Substring(B.[RowLog
Contents 0],19+14,2))))
    And A.[Link ID]=B.[Link ID]
    LEFT Join @ModifiedRawData C On C.[Page
IDS]=Convert(int,Convert(Varbinary(Max),Reverse(Substring(B.[RowLog
Contents 0],27+14,2))))
    And C.[File
IDS]=Convert(int,Convert(Varbinary(Max),Reverse(Substring(B.[RowLog
Contents 0],31+14,2))))
    And C.[Link ID]=B.[Link ID]
    Where  (A.[RowLog Contents 0] IS NOT NULL OR C.[RowLog Contents 0] IS
NOT NULL)


    Update B Set B.[RowLog Contents 0] =
    (CASE WHEN A.[RowLog Contents 0] IS NOT NULL AND C.[RowLog Contents
0] IS NOT NULL THEN  A.[RowLog Contents 0]+C.[RowLog Contents 0]
        WHEN A.[RowLog Contents 0] IS NULL AND C.[RowLog Contents 0] IS
NOT NULL THEN  C.[RowLog Contents 0]
        WHEN A.[RowLog Contents 0] IS NOT NULL AND C.[RowLog Contents 0]
IS NULL THEN  A.[RowLog Contents 0]
        END)
    --,B.[Update]=ISNULL(B.[Update],0)+1
    from @ModifiedRawData B
    LEFT Join @ModifiedRawData A On A.[Page
IDS]=Convert(int,Convert(Varbinary(Max),Reverse(Substring(B.[RowLog
Contents 0],15+14,2))))
    And A.[File
IDS]=Convert(int,Convert(Varbinary(Max),Reverse(Substring(B.[RowLog
Contents 0],19+14,2))))
    And A.[Link ID]<>B.[Link ID] And B.[Update]=0
    LEFT Join @ModifiedRawData C On C.[Page
IDS]=Convert(int,Convert(Varbinary(Max),Reverse(Substring(B.[RowLog
Contents 0],27+14,2))))
    And C.[File
IDS]=Convert(int,Convert(Varbinary(Max),Reverse(Substring(B.[RowLog
Contents 0],31+14,2))))
    And C.[Link ID]<>B.[Link ID] And B.[Update]=0
    Where  (A.[RowLog Contents 0] IS NOT NULL OR C.[RowLog Contents 0] IS
NOT NULL)

    UPDATE @ModifiedRawData  SET [RowLog Contents 0] =
    (Case When [RowLog Len]>=8000 Then
    Substring([RowLog Contents 0] ,15,[RowLog Len])
    When [RowLog Len]<8000 Then
    SUBSTRING([RowLog Contents
0],15+6,Convert(int,Convert(varbinary(max),REVERSE(Substring([RowLog
Contents 0],15,6)))))
    End)
    FROM @ModifiedRawData Where [LINK ID]=0

    UPDATE @ColumnNameAndData SET [hex_Value]=[RowLog Contents 0]
    --,A.[Update]=A.[Update]+1
    FROM @ColumnNameAndData A
```

```
    INNER JOIN @ModifiedRawData B ON

Convert(int,Convert(Varbinary(Max),Reverse(Substring([hex_value],17,4))))
=[PAGE IDS]
    AND  Convert(int,Substring([hex_value],9,2)) =B.[Link ID]
    Where [System_Type_Id] In (99,167,175,231,239,241,165,98) And [Link
ID] <>0

    UPDATE @ColumnNameAndData SET [hex_Value]=
    (CASE WHEN B.[RowLog Contents 0] IS NOT NULL AND C.[RowLog Contents
0] IS NOT NULL THEN  B.[RowLog Contents 0]+C.[RowLog Contents 0]
    WHEN B.[RowLog Contents 0] IS NULL AND C.[RowLog Contents 0] IS NOT
NULL THEN  C.[RowLog Contents 0]
    WHEN B.[RowLog Contents 0] IS NOT NULL AND C.[RowLog Contents 0] IS
NULL THEN  B.[RowLog Contents 0]
    END)
    --,A.[Update]=A.[Update]+1
    FROM @ColumnNameAndData A
    LEFT JOIN @ModifiedRawData B ON

Convert(int,Convert(Varbinary(Max),Reverse(Substring([hex_value],5,4))))=
B.[PAGE IDS]  And B.[Link ID] =0
    LEFT JOIN @ModifiedRawData C ON

Convert(int,Convert(Varbinary(Max),Reverse(Substring([hex_value],17,4))))
=C.[PAGE IDS]  And C.[Link ID] =0
    Where [System_Type_Id] In (99,167,175,231,239,241,165,98)  And
(B.[RowLog Contents 0] IS NOT NULL OR C.[RowLog Contents 0] IS NOT NULL)

    UPDATE @ColumnNameAndData SET [hex_Value]=[RowLog Contents 0]
    --,A.[Update]=A.[Update]+1
    FROM @ColumnNameAndData A
    INNER JOIN @ModifiedRawData B ON

Convert(int,Convert(Varbinary(Max),Reverse(Substring([hex_value],9,4))))=
[PAGE IDS]
    And Convert(int,Substring([hex_value],3,2))=[Link ID]
    Where [System_Type_Id] In (35,34,99) And [Link ID] <>0

    UPDATE @ColumnNameAndData SET [hex_Value]=[RowLog Contents 0]
    --,A.[Update]=A.[Update]+10
    FROM @ColumnNameAndData A
    INNER JOIN @ModifiedRawData B ON

Convert(int,Convert(Varbinary(Max),Reverse(Substring([hex_value],9,4))))=
[PAGE IDS]
    Where [System_Type_Id] In (35,34,99) And [Link ID] =0

    UPDATE @ColumnNameAndData SET [hex_Value]=[RowLog Contents 0]
    --,A.[Update]=A.[Update]+1
    FROM @ColumnNameAndData A
    INNER JOIN @ModifiedRawData B ON

Convert(int,Convert(Varbinary(Max),Reverse(Substring([hex_value],15,4))))
=[PAGE IDS]
    Where [System_Type_Id] In (35,34,99) And [Link ID] =0
```

```
    Update @ColumnNameAndData set [hex_value]= 0xFFFE +
Substring([hex_value],9,LEN([hex_value]))
    --,[Update]=[Update]+1
    Where [system_type_id]=241

CREATE TABLE [#temp_Data]
(
    [FieldName]  VARCHAR(MAX),
    [FieldValue] NVARCHAR(MAX),
    [Rowlogcontents] VARBINARY(8000),
    [Row ID] int
)

INSERT INTO #temp_Data
SELECT NAME,
CASE
 WHEN system_type_id IN (231, 239) THEN
LTRIM(RTRIM(CONVERT(NVARCHAR(max),hex_Value)))  --NVARCHAR ,NCHAR
 WHEN system_type_id IN (167,175) THEN
LTRIM(RTRIM(CONVERT(VARCHAR(max),hex_Value)))  --VARCHAR,CHAR
 WHEN system_type_id IN (35) THEN
LTRIM(RTRIM(CONVERT(VARCHAR(max),hex_Value))) --Text
 WHEN system_type_id IN (99) THEN
LTRIM(RTRIM(CONVERT(NVARCHAR(max),hex_Value))) --nText
 WHEN system_type_id = 48 THEN CONVERT(VARCHAR(MAX), CONVERT(TINYINT,
CONVERT(BINARY(1), REVERSE (hex_Value)))) --TINY INTEGER
 WHEN system_type_id = 52 THEN CONVERT(VARCHAR(MAX), CONVERT(SMALLINT,
CONVERT(BINARY(2), REVERSE (hex_Value)))) --SMALL INTEGER
 WHEN system_type_id = 56 THEN CONVERT(VARCHAR(MAX), CONVERT(INT,
CONVERT(BINARY(4), REVERSE(hex_Value)))) -- INTEGER
 WHEN system_type_id = 127 THEN CONVERT(VARCHAR(MAX), CONVERT(BIGINT,
CONVERT(BINARY(8), REVERSE(hex_Value))))-- BIG INTEGER
 WHEN system_type_id = 61 Then
CONVERT(VARCHAR(MAX),CONVERT(DATETIME,CONVERT(VARBINARY(8000),REVERSE
(hex_Value))),100) --DATETIME
 WHEN system_type_id =58 Then
CONVERT(VARCHAR(MAX),CONVERT(SMALLDATETIME,CONVERT(VARBINARY(8000),REVERS
E(hex_Value))),100) --SMALL DATETIME
 WHEN system_type_id = 108 THEN
CONVERT(VARCHAR(MAX),CONVERT(NUMERIC(38,20),
CONVERT(VARBINARY,CONVERT(VARBINARY(1),xprec)+CONVERT(VARBINARY(1),xscale
))+CONVERT(VARBINARY(1),0) + hex_Value)) --- NUMERIC
 WHEN system_type_id =106 THEN CONVERT(VARCHAR(MAX),
CONVERT(DECIMAL(38,20),
CONVERT(VARBINARY,Convert(VARBINARY(1),xprec)+CONVERT(VARBINARY(1),xscale
))+CONVERT(VARBINARY(1),0) + hex_Value)) --- DECIMAL
 WHEN system_type_id In(60,122) THEN
CONVERT(VARCHAR(MAX),Convert(MONEY,Convert(VARBINARY(8000),Reverse(hex_Va
lue))),2) --MONEY,SMALLMONEY
 WHEN system_type_id = 104 THEN CONVERT(VARCHAR(MAX),CONVERT
(BIT,CONVERT(BINARY(1), hex_Value)%2))  -- BIT
 WHEN system_type_id =62 THEN
RTRIM(LTRIM(STR(CONVERT(FLOAT,SIGN(CAST(CONVERT(VARBINARY(8000),Reverse(h
ex_Value)) AS BIGINT)) * (1.0 +
(CAST(CONVERT(VARBINARY(8000),Reverse(hex_Value)) AS BIGINT) &
0x000FFFFFFFFFFFFF) * POWER(CAST(2 AS FLOAT), -52)) * POWER(CAST(2 AS
FLOAT),((CAST(CONVERT(VARBINARY(8000),Reverse(hex_Value)) AS BIGINT) &
```

```
0x7ff0000000000000) / EXP(52 * LOG(2))-1023))),53,LEN(hex_Value)))) ---
FLOAT
 When system_type_id =59 THEN
Left(LTRIM(STR(CAST(SIGN(CAST(Convert(VARBINARY(8000),REVERSE(hex_Value))
AS BIGINT))* (1.0 + (CAST(CONVERT(VARBINARY(8000),Reverse(hex_Value)) AS
BIGINT) & 0x007FFFFF) * POWER(CAST(2 AS Real), -23)) * POWER(CAST(2 AS
Real),(((CAST(CONVERT(VARBINARY(8000),Reverse(hex_Value)) AS INT) )&
0x7f800000)/ EXP(23 * LOG(2))-127))AS REAL),23,23)),8) --Real
 WHEN system_type_id In (165,173) THEN (CASE WHEN CHARINDEX(0x,cast('' AS
XML).value('xs:hexBinary(sql:column("hex_Value"))', 'VARBINARY(8000)')) =
0 THEN '0x' ELSE '' END) +cast('' AS
XML).value('xs:hexBinary(sql:column("hex_Value"))', 'varchar(max)') --
BINARY,VARBINARY
 WHEN system_type_id =34 THEN (CASE WHEN CHARINDEX(0x,cast('' AS
XML).value('xs:hexBinary(sql:column("hex_Value"))', 'VARBINARY(8000)')) =
0 THEN '0x' ELSE '' END) +cast('' AS
XML).value('xs:hexBinary(sql:column("hex_Value"))', 'varchar(max)')  --
IMAGE
 WHEN system_type_id =36 THEN
CONVERT(VARCHAR(MAX),CONVERT(UNIQUEIDENTIFIER,hex_Value)) --
UNIQUEIDENTIFIER
 WHEN system_type_id =231 THEN
CONVERT(VARCHAR(MAX),CONVERT(sysname,hex_Value)) --SYSNAME
 WHEN system_type_id =241 THEN
CONVERT(VARCHAR(MAX),CONVERT(xml,hex_Value)) --XML

 WHEN system_type_id =189 THEN (CASE WHEN CHARINDEX(0x,cast('' AS
XML).value('xs:hexBinary(sql:column("hex_Value"))', 'VARBINARY(8000)')) =
0 THEN '0x' ELSE '' END) +cast('' AS
XML).value('xs:hexBinary(sql:column("hex_Value"))', 'varchar(max)') --
TIMESTAMP
 WHEN system_type_id=98 THEN (CASE
 WHEN CONVERT(INT,SUBSTRING(hex_Value,1,1))=56 THEN CONVERT(VARCHAR(MAX),
CONVERT(INT, CONVERT(BINARY(4),
REVERSE(Substring(hex_Value,3,Len(hex_Value))))))  -- INTEGER
 WHEN CONVERT(INT,SUBSTRING(hex_Value,1,1))=108 THEN
CONVERT(VARCHAR(MAX),CONVERT(numeric(38,20),CONVERT(VARBINARY(1),Substrin
g(hex_Value,3,1))
+CONVERT(VARBINARY(1),Substring(hex_Value,4,1))+CONVERT(VARBINARY(1),0) +
Substring(hex_Value,5,Len(hex_Value)))) --- NUMERIC
 WHEN CONVERT(INT,SUBSTRING(hex_Value,1,1))=167 THEN
LTRIM(RTRIM(CONVERT(VARCHAR(max),Substring(hex_Value,9,Len(hex_Value)))))
--VARCHAR,CHAR
 WHEN CONVERT(INT,SUBSTRING(hex_Value,1,1))=36 THEN
CONVERT(VARCHAR(MAX),CONVERT(UNIQUEIDENTIFIER,Substring((hex_Value),3,20)
)) --UNIQUEIDENTIFIER
 WHEN CONVERT(INT,SUBSTRING(hex_Value,1,1))=61 THEN
CONVERT(VARCHAR(MAX),CONVERT(DATETIME,CONVERT(VARBINARY(8000),REVERSE
(Substring(hex_Value,3,LEN(hex_Value)) )),100) --DATETIME
 WHEN CONVERT(INT,SUBSTRING(hex_Value,1,1))=165 THEN '0x'+
SUBSTRING((CASE WHEN CHARINDEX(0x,cast('' AS
XML).value('xs:hexBinary(sql:column("hex_Value"))', 'VARBINARY(8000)')) =
0 THEN '0x' ELSE '' END) +cast('' AS
XML).value('xs:hexBinary(sql:column("hex_Value"))',
'varchar(max)'),11,LEN(hex_Value)) -- BINARY,VARBINARY
 END)
```

```
END AS FieldValue
,[Rowlogcontents]
,[Row ID]
FROM @ColumnNameAndData ORDER BY nullbit

--Create the column name in the same order to do pivot table.

DECLARE @FieldName VARCHAR(max)
SET @FieldName = STUFF(
(
    SELECT ',' + CAST(QUOTENAME([Name]) AS VARCHAR(MAX)) FROM syscolumns
WHERE id=object_id('' + @SchemaName_n_TableName + '')
    FOR XML PATH('')), 1, 1, '')

--Finally did pivot table and get the data back in the same format.

SET @sql = 'SELECT ' + @FieldName  + ' FROM #temp_Data PIVOT
(Min([FieldValue]) FOR FieldName IN (' + @FieldName  + ')) AS pvt'
EXEC sp_executesql @sql
```

۹-۲ پیوست ب

```
Create PROCEDURE Recover_Modified_Data_Proc
@Database_Name NVARCHAR(MAX),
@SchemaName_n_TableName NVARCHAR(MAX),
@Date_From datetime='1900/01/01',
@Date_To datetime ='9999/12/31'
AS
DECLARE @parms nvarchar(1024)
DECLARE @Fileid INT
DECLARE @Pageid INT
DECLARE @Slotid INT
DECLARE @RowLogContents0 VARBINARY(8000)
DECLARE @RowLogContents1 VARBINARY(8000)
DECLARE @RowLogContents3 VARBINARY(8000)
DECLARE @RowLogContents3_Var VARCHAR(MAX)

DECLARE @RowLogContents4 VARBINARY(8000)
DECLARE @LogRecord VARBINARY(8000)
DECLARE @LogRecord_Var VARCHAR(MAX)

DECLARE @ConsolidatedPageID VARCHAR(MAX)
Declare @AllocUnitID as bigint
Declare @TransactionID as VARCHAR(MAX)
Declare @Operation as VARCHAR(MAX)
Declare @DatabaseCollation VARCHAR(MAX)


/*  Pick The actual data
*/
declare @temppagedata table
(
[ParentObject] sysname,
```

```
[Object] sysname,
[Field] sysname,
[Value] sysname)

declare @pagedata table
(
[Page ID] sysname,
[AllocUnitId] bigint,
[ParentObject] sysname,
[Object] sysname,
[Field] sysname,
[Value] sysname)


    DECLARE Page_Data_Cursor CURSOR FOR
    /*We need to filter LOP_MODIFY_ROW,LOP_MODIFY_COLUMNS from log for
modified records & Get its Slot No, Page ID & AllocUnit ID*/
    SELECT [PAGE ID],[Slot ID],[AllocUnitId]
    FROM    sys.fn_dblog(NULL, NULL)
    WHERE
    AllocUnitId IN
    (Select [Allocation_unit_id] from sys.allocation_units allocunits
    INNER JOIN sys.partitions partitions ON (allocunits.type IN (1, 3)
    AND partitions.hobt_id = allocunits.container_id) OR (allocunits.type
= 2
    AND partitions.partition_id = allocunits.container_id)
    Where object_id=object_ID('' + @SchemaName_n_TableName + ''))
    AND Operation IN ('LOP_MODIFY_ROW','LOP_MODIFY_COLUMNS')  AND
[Context] IN ('LCX_HEAP','LCX_CLUSTERED')
    /*Use this subquery to filter the date*/

    AND [TRANSACTION ID] IN (SELECT DISTINCT [TRANSACTION ID] FROM
sys.fn_dblog(NULL, NULL)
    WHERE Context IN ('LCX_NULL') AND Operation in ('LOP_BEGIN_XACT')
    AND [Transaction Name]='UPDATE'
    AND  CONVERT(NVARCHAR(11),[Begin Time]) BETWEEN @Date_From AND
@Date_To)

    /*****************************************/

    GROUP BY [PAGE ID],[Slot ID],[AllocUnitId]
    ORDER BY [Slot ID]

    OPEN Page_Data_Cursor

    FETCH NEXT FROM Page_Data_Cursor INTO @ConsolidatedPageID,
@Slotid,@AllocUnitID

    WHILE @@FETCH_STATUS = 0
    BEGIN
        DECLARE @hex_pageid AS VARCHAR(Max)
        /*Page ID contains File Number and page number It looks like
0001:00000130.
          In this example 0001 is file Number &  00000130 is Page Number
& These numbers are in Hex format*/
        SET
@Fileid=SUBSTRING(@ConsolidatedPageID,0,CHARINDEX(':',@ConsolidatedPageID
```

```
)) -- Seperate File ID from Page ID
        SET @hex_pageid ='0x'+
SUBSTRING(@ConsolidatedPageID,CHARINDEX(':',@ConsolidatedPageID)+1,Len(@C
onsolidatedPageID))  ---Seperate the page ID

        SELECT @Pageid=Convert(INT,cast('' AS
XML).value('xs:hexBinary(substring(sql:variable("@hex_pageid"),sql:column
("t.pos")) )', 'varbinary(max)')) -- Convert Page ID from hex to integer
        FROM (SELECT CASE substring(@hex_pageid, 1, 2) WHEN '0x' THEN 3
ELSE 0 END) AS t(pos)

        DELETE @temppagedata
        -- Now we need to get the actual data (After modification) from
the page
        INSERT INTO @temppagedata EXEC( 'DBCC PAGE(' + @DataBase_Name +
', ' + @fileid + ', ' + @pageid + ', 3) with tableresults,no_infomsgs;');
        -- Add Page Number and allocUnit ID in data to identity which one
page it belongs to.
        INSERT INTO @pagedata SELECT
@ConsolidatedPageID,@AllocUnitID,[ParentObject],[Object],[Field] ,[Value]
FROM @temppagedata

        FETCH NEXT FROM Page_Data_Cursor INTO  @ConsolidatedPageID,
@Slotid,@AllocUnitID
    END

CLOSE Page_Data_Cursor
DEALLOCATE Page_Data_Cursor

DECLARE @Newhexstring VARCHAR(MAX);


DECLARE @ModifiedRawData TABLE
(
  [ID] INT IDENTITY(1,1),
  [PAGE ID] VARCHAR(MAX),
  [Slot ID] INT,
  [AllocUnitId] BIGINT,
  [RowLog Contents 0_var] VARCHAR(MAX),
  [RowLog Contents 0] VARBINARY(8000)
)
--The modified data is in multiple rows in the page, so we need to
convert it into one row as a single hex value.
--This hex value is in string format
INSERT INTO @ModifiedRawData ([PAGE ID],[Slot ID],[AllocUnitId]
,[RowLog Contents 0_var])
SELECT B.[PAGE ID],A.[Slot ID],A.[AllocUnitId]
,(
SELECT REPLACE(STUFF((SELECT
REPLACE(SUBSTRING([VALUE],CHARINDEX(':',[Value])+1,48),'†','')
FROM @pagedata C  WHERE B.[Page ID]= C.[Page ID] And A.[Slot ID]
=LTRIM(RTRIM(SUBSTRING(C.[ParentObject],5,3))) And [Object] Like '%Memory
Dump%'
Group By [Value] FOR XML PATH('') ),1,1,'') ,' ','')
) AS [Value]

FROM sys.fn_dblog(NULL, NULL) A
```

```
INNER JOIN @pagedata B On A.[PAGE ID]=B.[PAGE ID]
AND A.[AllocUnitId]=B.[AllocUnitId]
AND A.[Slot ID] =LTRIM(RTRIM(Substring(B.[ParentObject],5,3)))
AND B.[Object] Like '%Memory Dump%'
WHERE A.AllocUnitId IN
(Select [Allocation_unit_id] from sys.allocation_units allocunits
INNER JOIN sys.partitions partitions ON (allocunits.type IN (1, 3)
AND partitions.hobt_id = allocunits.container_id) OR (allocunits.type = 2
AND partitions.partition_id = allocunits.container_id)
Where object_id=object_ID('' + @SchemaName_n_TableName + ''))
AND Operation in ('LOP_MODIFY_ROW','LOP_MODIFY_COLUMNS') AND [Context] IN
('LCX_HEAP','LCX_CLUSTERED')
/*Use this subquery to filter the date*/

AND [TRANSACTION ID] IN (Select DISTINCT [TRANSACTION ID] FROM
sys.fn_dblog(NULL, NULL)
Where Context IN ('LCX_NULL') AND Operation IN ('LOP_BEGIN_XACT')
AND [Transaction Name]='UPDATE'
AND  CONVERT(NVARCHAR(11),[Begin Time]) BETWEEN @Date_From AND @Date_To)

/*****************************************/
GROUP BY B.[PAGE ID],A.[Slot ID],A.[AllocUnitId]--,[Transaction ID]
ORDER BY [Slot ID]


-- Convert the hex value data in string, convert it into Hex value as
well.
UPDATE @ModifiedRawData  SET [RowLog Contents 0] = cast('' AS
XML).value('xs:hexBinary(substring(sql:column("[RowLog Contents 0_var]"),
0) )', 'varbinary(max)')
FROM @ModifiedRawData

---Now we have modifed data plus its slot ID , page ID and allocunit as
well.
--After that we need to get the old values before modfication, these
datas are in chunks.
DECLARE Page_Data_Cursor CURSOR FOR

Select [PAGE ID],[Slot ID],[AllocUnitId],[Transaction ID],[RowLog
Contents 0], [RowLog Contents 1],[RowLog Contents 3],[RowLog Contents 4]
,Substring ([Log Record],[Log Record Fixed Length],([Log Record
Length]+1)-([Log Record Fixed Length])) as [Log Record]
,Operation
FROM    sys.fn_dblog(NULL, NULL)
WHERE   AllocUnitId IN
(Select [Allocation_unit_id] from sys.allocation_units allocunits
INNER JOIN sys.partitions partitions ON (allocunits.type IN (1, 3)
AND partitions.hobt_id = allocunits.container_id) OR (allocunits.type = 2
AND partitions.partition_id = allocunits.container_id)
Where object_id=object_ID('' + @SchemaName_n_TableName + ''))
AND Operation in ('LOP_MODIFY_ROW','LOP_MODIFY_COLUMNS') And [Context] IN
('LCX_HEAP','LCX_CLUSTERED')
/*Use this sub query to filter the date*/

AND [TRANSACTION ID] IN (Select DISTINCT [TRANSACTION ID] FROM
sys.fn_dblog(NULL, NULL)
WHERE Context IN ('LCX_NULL') AND Operation IN ('LOP_BEGIN_XACT')
```

```
AND [Transaction Name]='UPDATE'
AND  CONVERT(NVARCHAR(11),[Begin Time]) BETWEEN @Date_From AND @Date_To)

/**************************************/
Order By [Slot ID],[Transaction ID] DESC

OPEN Page_Data_Cursor

    FETCH NEXT FROM Page_Data_Cursor INTO @ConsolidatedPageID,
@Slotid,@AllocUnitID,@TransactionID,@RowLogContents0,@RowLogContents1,@Ro
wLogContents3,@RowLogContents4,@LogRecord,@Operation
    WHILE @@FETCH_STATUS = 0
    BEGIN
           IF @Operation ='LOP_MODIFY_ROW'
              BEGIN
                       /* If it is @Operation Type is 'LOP_MODIFY_ROW'
then it is very simple to recover the modified data. The old data is in
[RowLog Contents 0] Field and modified data is in [RowLog Contents 1]
Field. Simply replace it with the modified data and get the old data.
                       */
                       INSERT INTO @ModifiedRawData ([PAGE ID],[Slot
ID],[AllocUnitId],[RowLog Contents 0_var])
                       SELECT TOP 1  @ConsolidatedPageID AS [PAGE
ID],@Slotid AS [Slot ID],@AllocUnitID AS [AllocUnitId]
                       ,REPLACE (UPPER([RowLog Contents
0_var]),UPPER(CAST('' AS
XML).value('xs:hexBinary(sql:variable("@RowLogContents1") )',
'varchar(max)')),UPPER(cast('' AS
XML).value('xs:hexBinary(sql:variable("@RowLogContents0") )',
'varchar(max)'))) AS [RowLog Contents 0_var]
                       FROM  @ModifiedRawData WHERE [PAGE
ID]=@ConsolidatedPageID And [Slot ID]=@Slotid And
[AllocUnitId]=@AllocUnitID ORDER BY [ID] DESC

                       --- Convert the old data which is in string format
to hex format.
                       UPDATE @ModifiedRawData  SET [RowLog Contents 0] =
cast('' AS XML).value('xs:hexBinary(substring(sql:column("[RowLog
Contents 0_var]"), 0) )', 'varbinary(max)')
                       FROM @ModifiedRawData Where [Slot ID]=@SlotID

              END
          IF @Operation ='LOP_MODIFY_COLUMNS'
              BEGIN

                          /* If it is @Operation Type is 'LOP_MODIFY_ROW'
then we need to follow a different procedure to recover modified
                             .Because this time the data is also in
chunks but merge with the data log.
                          */
                            --First, we need to get the [RowLog Contents
3] Because in [Log Record] field the modified data is available after the
[RowLog Contents 3] data.
                             SET @RowLogContents3_Var=cast('' AS
XML).value('xs:hexBinary(sql:variable("@RowLogContents3") )',
'varchar(max)')
                             SET @LogRecord_Var =cast('' AS
```

```
XML).value('xs:hexBinary(sql:variable("@LogRecord"))', 'varchar(max)')

                                DECLARE @RowLogData_Var VARCHAR(Max)
                                DECLARE @RowLogData_Hex VARBINARY(Max)
                                ---First get the modifed data chunks in
string format
                                SET @RowLogData_Var =
SUBSTRING(@LogRecord_Var, CHARINDEX(@RowLogContents3_Var,@LogRecord_Var)
+LEN(@RowLogContents3_Var) ,LEN(@LogRecord_Var))
                                --Then convert it into the hex values.
                                SELECT @RowLogData_Hex=CAST('' AS
XML).value('xs:hexBinary( substring(sql:variable("@RowLogData_Var"),0)
)', 'varbinary(max)')
                                FROM (SELECT CASE SUBSTRING(@RowLogData_Var,
1, 2) WHEN '0x' THEN 3 ELSE 0 END) AS t(pos)
                                DECLARE @TotalFixedLengthData INT
                                DECLARE @FixedLength_Offset INT
                                DECLARE @VariableLength_Offset INT
                                DECLARE @VariableLength_Offset_Start INT
                                DECLARE @VariableLengthIncrease INT
                                DECLARE @FixedLengthIncrease INT
                                DECLARE @OldFixedLengthStartPosition INT
                                DECLARE @FixedLength_Loc INT
                                DECLARE @VariableLength_Loc INT
                                DECLARE @FixedOldValues VARBINARY(MAX)
                                DECLARE @FixedNewValues VARBINARY(MAX)
                                DECLARE @VariableOldValues VARBINARY(MAX)
                                DECLARE @VariableNewValues VARBINARY(MAX)

                                -- Before recovering the modfied data we need
to get the total fixed length data size and start position of the
varaible data

                                SELECT TOP 1
@TotalFixedLengthData=CONVERT(SMALLINT, CONVERT(BINARY(2),
REVERSE(SUBSTRING([RowLog Contents 0] , 2 + 1, 2))))

,@VariableLength_Offset_Start=CONVERT(SMALLINT, CONVERT(BINARY(2),
REVERSE(SUBSTRING([RowLog Contents 0] , 2 + 1, 2))))+5+CONVERT(INT,
ceiling(CONVERT(INT, CONVERT(BINARY(2), REVERSE(SUBSTRING([RowLog
Contents 0] , CONVERT(SMALLINT, CONVERT(BINARY(2),
REVERSE(SUBSTRING([RowLog Contents 0] , 2 + 1, 2))) + 1, 2))))/8.0))
                                FROM @ModifiedRawData
                                ORDER BY [ID] DESC

                                SET @FixedLength_Offset=
CONVERT(BINARY(2),REVERSE(CONVERT(BINARY(4),(@RowLogContents0))))--)
                                SET
@VariableLength_Offset=CONVERT(int,CONVERT(BINARY(2),REVERSE(@RowLogConte
nts0)))

                                /* We already have modified data chunks in
@RowLogData_Hex but this data is in merge format (modified plus actual
data)
                                  So , here we need [Row Log Contents 1]
field , because in this field we have the data length both the modified
and actual data
```

```
                                so this length will help us to break it
into original and modified data chunks.
                                */
                                SET @FixedLength_Loc=
CONVERT(INT,SUBSTRING(@RowLogContents1,1,1))
                                SET @VariableLength_Loc
=CONVERT(INT,SUBSTRING(@RowLogContents1,3,1))

                                /*First , we need to break Fix length data
actual with the help of data length  */
                                SET @OldFixedLengthStartPosition=
CHARINDEX(@RowLogContents4,@RowLogData_Hex)
                                SET @FixedOldValues =
SUBSTRING(@RowLogData_Hex,@OldFixedLengthStartPosition,@FixedLength_Loc)
                                SET @FixedLengthIncrease = (CASE WHEN
(Len(@FixedOldValues)%4)=0 THEN 1 ELSE (4-(LEN(@FixedOldValues)%4))  END)
                                /*After that , we need to break Fix length
data modified data with the help of data length  */
                                SET @FixedNewValues
=SUBSTRING(@RowLogData_Hex,@OldFixedLengthStartPosition+@FixedLength_Loc+
@FixedLengthIncrease,@FixedLength_Loc)

                                /*Same we need to break the variable data
with the help of data length*/
                                SET @VariableOldValues
=SUBSTRING(@RowLogData_Hex,@OldFixedLengthStartPosition+@FixedLength_Loc+
@FixedLengthIncrease+@FixedLength_Loc+(@FixedLengthIncrease),@VariableLen
gth_Loc)
                                SET @VariableLengthIncrease =  (CASE WHEN
(LEN(@VariableOldValues)%4)=0 THEN 1 ELSE (4-
(Len(@VariableOldValues)%4))+1  END)
                                SET @VariableOldValues =(Case When
@VariableLength_Loc =1 Then  @VariableOldValues+0x00 else
@VariableOldValues end)

                                SET @VariableNewValues
=SUBSTRING(SUBSTRING(@RowLogData_Hex,@OldFixedLengthStartPosition+@FixedL
ength_Loc+@FixedLengthIncrease+@FixedLength_Loc+(@FixedLengthIncrease-
1)+@VariableLength_Loc+@VariableLengthIncrease,Len(@RowLogData_Hex)+1),1,
Len(@RowLogData_Hex)+1) --LEN(@VariableOldValues)

                                /*here we need to replace the fixed length &
variable length actaul data with modifed data
                                */

                                Select top 1 @VariableNewValues=Case
                                When
Charindex(Substring(@VariableNewValues,0,Len(@VariableNewValues)+1),[RowL
og Contents 0])<>0 Then
Substring(@VariableNewValues,0,Len(@VariableNewValues)+1)
                                When
Charindex(Substring(@VariableNewValues,0,Len(@VariableNewValues)),[RowLog
Contents 0])<>0 Then
Substring(@VariableNewValues,0,Len(@VariableNewValues))
                                When
Charindex(Substring(@VariableNewValues,0,Len(@VariableNewValues)-
1),[RowLog Contents 0])<>0 Then
```

```
Substring(@VariableNewValues,0,Len(@VariableNewValues)-1)--3 --
Substring(@VariableNewValues,0,Len(@VariableNewValues)-1)
                              When
Charindex(Substring(@VariableNewValues,0,Len(@VariableNewValues)-
2),[RowLog Contents 0])<>0 Then
Substring(@VariableNewValues,0,Len(@VariableNewValues)-2)
                              When
Charindex(Substring(@VariableNewValues,0,Len(@VariableNewValues)-
3),[RowLog Contents 0])<>0 Then
Substring(@VariableNewValues,0,Len(@VariableNewValues)-3) --5--
Substring(@VariableNewValues,0,Len(@VariableNewValues)-3)
                              End
                              FROM @ModifiedRawData  Where [Slot
ID]=@SlotID  ORDER BY [ID] DESC

                              INSERT INTO @ModifiedRawData ([PAGE ID],[Slot
ID],[AllocUnitId],[RowLog Contents 0_var],[RowLog Contents 0])
                              SELECT TOP 1  @ConsolidatedPageID AS [PAGE
ID],@Slotid AS [Slot ID],@AllocUnitID AS [AllocUnitId],NULL
                              ,CAST(REPLACE(SUBSTRING([RowLog Contents
0],0,@TotalFixedLengthData+1),@FixedNewValues, @FixedOldValues) AS
VARBINARY(max))
                              + SUBSTRING([RowLog Contents 0],
@TotalFixedLengthData + 1, 2)
                              + SUBSTRING([RowLog Contents 0],
@TotalFixedLengthData + 3, CONVERT(INT, ceiling(CONVERT(INT,
CONVERT(BINARY(2), REVERSE(SUBSTRING([RowLog Contents 0],
@TotalFixedLengthData + 1, 2))))/8.0)))
                              + SUBSTRING([RowLog Contents 0],
@TotalFixedLengthData + 3 + CONVERT(INT, ceiling(CONVERT(INT,
CONVERT(BINARY(2), REVERSE(SUBSTRING([RowLog Contents 0],
@TotalFixedLengthData + 1, 2))))/8.0)), 2)
                              + Substring([RowLog Contents
0],@VariableLength_Offset_Start,(@VariableLength_Offset-
(@VariableLength_Offset_Start-1)))
                              + CAST(REPLACE(SUBSTRING([RowLog Contents
0],@VariableLength_Offset+1,Len(@VariableNewValues))
                              , @VariableNewValues
                              , @VariableOldValues) AS VARBINARY)
                              + Substring([RowLog Contents
0],@VariableLength_Offset+Len(@VariableNewValues)+1,LEN([RowLog Contents
0]))
                              FROM @ModifiedRawData  Where [Slot
ID]=@SlotID  ORDER BY [ID] DESC

              END

        FETCH NEXT FROM Page_Data_Cursor INTO   @ConsolidatedPageID,
@Slotid,@AllocUnitID,@TransactionID,@RowLogContents0,@RowLogContents1,@Ro
wLogContents3,@RowLogContents4,@LogRecord,@Operation
    END

CLOSE Page_Data_Cursor
DEALLOCATE Page_Data_Cursor

DECLARE @RowLogContents VARBINARY(8000)
Declare @AllocUnitName NVARCHAR(Max)
```

```
Declare @SQL NVARCHAR(Max)

DECLARE @bitTable TABLE
(
  [ID] INT,
  [Bitvalue] INT
)
----Create table to set the bit position of one byte.

INSERT INTO @bitTable
SELECT 0,2 UNION ALL
SELECT 1,2 UNION ALL
SELECT 2,4 UNION ALL
SELECT 3,8 UNION ALL
SELECT 4,16 UNION ALL
SELECT 5,32 UNION ALL
SELECT 6,64 UNION ALL
SELECT 7,128

--Create table to collect the row data.
DECLARE @DeletedRecords TABLE
(
    [ID] INT IDENTITY(1,1),
    [RowLogContents]    VARBINARY(8000),
    [AllocUnitID]       BIGINT,
    [Transaction ID]    NVARCHAR(Max),
    [Slot ID]           INT,
    [FixedLengthData]   SMALLINT,
    [TotalNoOfCols]     SMALLINT,
    [NullBitMapLength]  SMALLINT,
    [NullBytes]         VARBINARY(8000),
    [TotalNoofVarCols]  SMALLINT,
    [ColumnOffsetArray] VARBINARY(8000),
    [VarColumnStart]    SMALLINT,
    [NullBitMap]        VARCHAR(MAX)
)
--Create a common table expression to get all the row data plus how many
bytes we have for each row.
;WITH RowData AS (
SELECT

[RowLog Contents 0] AS [RowLogContents]

,@AllocUnitID AS [AllocUnitID]

,[ID] AS [Transaction ID]

,[Slot ID] as [Slot ID]
--[Fixed Length Data] = Substring (RowLog content 0, Status Bit A+ Status
Bit B + 1,2 bytes)
,CONVERT(SMALLINT, CONVERT(BINARY(2), REVERSE(SUBSTRING([RowLog Contents
0], 2 + 1, 2)))) AS [FixedLengthData]  --@FixedLengthData

 --[TotalnoOfCols] =  Substring (RowLog content 0, [Fixed Length Data] +
1,2 bytes)
,CONVERT(INT, CONVERT(BINARY(2), REVERSE(SUBSTRING([RowLog Contents 0],
CONVERT(SMALLINT, CONVERT(BINARY(2)
```

```
,REVERSE(SUBSTRING([RowLog Contents 0], 2 + 1, 2)))) + 1, 2)))) as
[TotalNoOfCols]

--[NullBitMapLength]=ceiling([Total No of Columns] /8.0)
,CONVERT(INT, ceiling(CONVERT(INT, CONVERT(BINARY(2),
REVERSE(SUBSTRING([RowLog Contents 0], CONVERT(SMALLINT,
CONVERT(BINARY(2)
,REVERSE(SUBSTRING([RowLog Contents 0], 2 + 1, 2)))) + 1, 2))))/8.0)) as
[NullBitMapLength]

--[Null Bytes] = Substring (RowLog content 0, Status Bit A+ Status Bit B
+ [Fixed Length Data] +1, [NullBitMapLength] )
,SUBSTRING([RowLog Contents 0], CONVERT(SMALLINT, CONVERT(BINARY(2),
REVERSE(SUBSTRING([RowLog Contents 0], 2 + 1, 2)))) + 3,
CONVERT(INT, ceiling(CONVERT(INT, CONVERT(BINARY(2),
REVERSE(SUBSTRING([RowLog Contents 0], CONVERT(SMALLINT,
CONVERT(BINARY(2)
,REVERSE(SUBSTRING([RowLog Contents 0], 2 + 1, 2)))) + 1, 2))))/8.0))) as
[NullBytes]

--[TotalNoofVarCols] = Substring (RowLog content 0, Status Bit A+ Status
Bit B + [Fixed Length Data] +1, [Null Bitmap length] + 2 )
,(CASE WHEN SUBSTRING([RowLog Contents 0], 1, 1) In (0x30,0x70) THEN
CONVERT(INT, CONVERT(BINARY(2), REVERSE(SUBSTRING([RowLog Contents 0],
CONVERT(SMALLINT, CONVERT(BINARY(2), REVERSE(SUBSTRING([RowLog Contents
0], 2 + 1, 2)))) + 3
+ CONVERT(INT, ceiling(CONVERT(INT, CONVERT(BINARY(2),
REVERSE(SUBSTRING([RowLog Contents 0], CONVERT(SMALLINT,
CONVERT(BINARY(2)
,REVERSE(SUBSTRING([RowLog Contents 0], 2 + 1, 2)))) + 1, 2))))/8.0)),
2))))  ELSE null  END) AS [TotalNoofVarCols]

--[ColumnOffsetArray]= Substring (RowLog content 0, Status Bit A+ Status
Bit B + [Fixed Length Data] +1, [Null Bitmap length] + 2 ,
[TotalNoofVarCols]*2 )
,(CASE WHEN SUBSTRING([RowLog Contents 0], 1, 1) In (0x30,0x70) THEN
SUBSTRING([RowLog Contents 0]
, CONVERT(SMALLINT, CONVERT(BINARY(2), REVERSE(SUBSTRING([RowLog Contents
0], 2 + 1, 2)))) + 3
+ CONVERT(INT, ceiling(CONVERT(INT, CONVERT(BINARY(2),
REVERSE(SUBSTRING([RowLog Contents 0], CONVERT(SMALLINT,
CONVERT(BINARY(2)
,REVERSE(SUBSTRING([RowLog Contents 0], 2 + 1, 2)))) + 1, 2))))/8.0)) + 2
, (CASE WHEN SUBSTRING([RowLog Contents 0], 1, 1) In (0x30,0x70) THEN
CONVERT(INT, CONVERT(BINARY(2), REVERSE(SUBSTRING([RowLog Contents 0],
CONVERT(SMALLINT, CONVERT(BINARY(2), REVERSE(SUBSTRING([RowLog Contents
0], 2 + 1, 2)))) + 3
+ CONVERT(INT, ceiling(CONVERT(INT, CONVERT(BINARY(2),
REVERSE(SUBSTRING([RowLog Contents 0], CONVERT(SMALLINT,
CONVERT(BINARY(2)
,REVERSE(SUBSTRING([RowLog Contents 0], 2 + 1, 2)))) + 1, 2))))/8.0)),
2))))  ELSE null  END)
* 2)  ELSE null  END) AS [ColumnOffsetArray]

--  Variable column Start = Status Bit A+ Status Bit B + [Fixed Length
Data] + [Null Bitmap length] + 2+([TotalNoofVarCols]*2)
,CASE WHEN SUBSTRING([RowLog Contents 0], 1, 1)In (0x30,0x70)
```

```
THEN  (
CONVERT(SMALLINT, CONVERT(BINARY(2), REVERSE(SUBSTRING([RowLog Contents
0], 2 + 1, 2)))) + 4

+ CONVERT(INT, ceiling(CONVERT(INT, CONVERT(BINARY(2),
REVERSE(SUBSTRING([RowLog Contents 0], CONVERT(SMALLINT,
CONVERT(BINARY(2)
,REVERSE(SUBSTRING([RowLog Contents 0], 2 + 1, 2)))) + 1, 2))))/8.0))

+ ((CASE WHEN SUBSTRING([RowLog Contents 0], 1, 1) In (0x30,0x70) THEN
CONVERT(INT, CONVERT(BINARY(2), REVERSE(SUBSTRING([RowLog Contents 0],
CONVERT(SMALLINT, CONVERT(BINARY(2), REVERSE(SUBSTRING([RowLog Contents
0], 2 + 1, 2)))) + 3
+ CONVERT(INT, ceiling(CONVERT(INT, CONVERT(BINARY(2),
REVERSE(SUBSTRING([RowLog Contents 0], CONVERT(SMALLINT,
CONVERT(BINARY(2)
,REVERSE(SUBSTRING([RowLog Contents 0], 2 + 1, 2)))) + 1, 2))))/8.0)),
2))))  ELSE null  END) * 2))

ELSE null End AS [VarColumnStart]
From @ModifiedRawData

),

---Use this technique to repeate the row till the no of bytes of the row.
N1 (n) AS (SELECT 1 UNION ALL SELECT 1),
N2 (n) AS (SELECT 1 FROM N1 AS X, N1 AS Y),
N3 (n) AS (SELECT 1 FROM N2 AS X, N2 AS Y),
N4 (n) AS (SELECT ROW_NUMBER() OVER(ORDER BY X.n)
           FROM N3 AS X, N3 AS Y)

insert into @DeletedRecords
Select   RowLogContents
        ,[AllocUnitID]
        ,[Transaction ID]
        ,[Slot ID]
        ,[FixedLengthData]
        ,[TotalNoOfCols]
        ,[NullBitMapLength]
        ,[NullBytes]
        ,[TotalNoofVarCols]
        ,[ColumnOffsetArray]
        ,[VarColumnStart]
         --Get the Null value against each column (1 means null zero
means not null)
        ,[NullBitMap]=(REPLACE(STUFF((SELECT ',' +
        (CASE WHEN [ID]=0 THEN CONVERT(NVARCHAR(1),(SUBSTRING(NullBytes,
n, 1) % 2))  ELSE CONVERT(NVARCHAR(1),((SUBSTRING(NullBytes, n, 1) /
[Bitvalue]) % 2)) END) --as [nullBitMap]
FROM
N4 AS Nums
Join RowData AS C ON n<=NullBitMapLength
Cross Join @bitTable WHERE C.[RowLogContents]=D.[RowLogContents] ORDER BY
[RowLogContents],n ASC FOR XML PATH('')),1,1,''),',','''))
FROM RowData D

CREATE TABLE [#temp_Data]
```

```
(

    [FieldName]  VARCHAR(MAX) COLLATE database_default NOT NULL,
    [FieldValue] VARCHAR(MAX) COLLATE database_default NULL,
    [Rowlogcontents] VARBINARY(8000),
    [Transaction ID] VARCHAR(MAX) COLLATE database_default NOT NULL,
    [Slot ID] INT,
    [NonID] INT,
    --[System_type_id] int

)
---Create common table expression and join it with the rowdata table
--to get each column details
;With CTE AS (
/*This part is for variable data columns*/
SELECT
A.[ID],
Rowlogcontents,
[Transaction ID],
[Slot ID],
NAME ,
cols.leaf_null_bit AS nullbit,
leaf_offset,
ISNULL(syscolumns.length, cols.max_length) AS [length],
cols.system_type_id,
cols.leaf_bit_position AS bitpos,
ISNULL(syscolumns.xprec, cols.precision) AS xprec,
ISNULL(syscolumns.xscale, cols.scale) AS xscale,
SUBSTRING([nullBitMap], cols.leaf_null_bit, 1) AS is_null,
--Calculate the variable column size from the variable column offset
array
(CASE WHEN leaf_offset<1 and SUBSTRING([nullBitMap], cols.leaf_null_bit,
1)=0 THEN
CONVERT(SMALLINT, CONVERT(BINARY(2), REVERSE (SUBSTRING
([ColumnOffsetArray], (2 * leaf_offset*-1) - 1, 2)))) ELSE 0 END) AS
[Column value Size],

---Calculate the column length
(CASE WHEN leaf_offset<1 and SUBSTRING([nullBitMap], cols.leaf_null_bit,
1)=0 THEN  CONVERT(SMALLINT, CONVERT(BINARY(2), REVERSE (SUBSTRING
([ColumnOffsetArray], (2 * (leaf_offset*-1)) - 1, 2))))
- ISNULL(NULLIF(CONVERT(SMALLINT, CONVERT(BINARY(2), REVERSE (SUBSTRING
([ColumnOffsetArray], (2 * ((leaf_offset*-1) - 1)) - 1, 2)))), 0),
[varColumnStart])
ELSE 0 END) AS [Column Length]

--Get the Hexa decimal value from the RowlogContent
--HexValue of the variable column=Substring([Column value Size] - [Column
Length] + 1,[Column Length])
--This is the data of your column but in the Hexvalue
,CASE WHEN SUBSTRING([nullBitMap], cols.leaf_null_bit, 1)=1 THEN NULL
ELSE
SUBSTRING(Rowlogcontents,((CASE WHEN leaf_offset<1 and
SUBSTRING([nullBitMap], cols.leaf_null_bit, 1)=0 THEN CONVERT(SMALLINT,
CONVERT(BINARY(2), REVERSE (SUBSTRING ([ColumnOffsetArray], (2 *
leaf_offset*-1) - 1, 2)))) ELSE 0 END)
- ((CASE WHEN leaf_offset<1 and SUBSTRING([nullBitMap],
```

```
cols.leaf_null_bit, 1)=0 THEN  CONVERT(SMALLINT, CONVERT(BINARY(2),
REVERSE (SUBSTRING ([ColumnOffsetArray], (2 * (leaf_offset*-1)) - 1,
2))))
- ISNULL(NULLIF(CONVERT(SMALLINT, CONVERT(BINARY(2), REVERSE (SUBSTRING
([ColumnOffsetArray], (2 * ((leaf_offset*-1) - 1)) - 1, 2)))), 0),
[varColumnStart])
ELSE 0 END))) + 1,((CASE WHEN leaf_offset<1 and SUBSTRING([nullBitMap],
cols.leaf_null_bit, 1)=0 THEN  CONVERT(SMALLINT, CONVERT(BINARY(2),
REVERSE (SUBSTRING ([ColumnOffsetArray], (2 * (leaf_offset*-1)) - 1,
2))))
- ISNULL(NULLIF(CONVERT(SMALLINT, CONVERT(BINARY(2), REVERSE (SUBSTRING
([ColumnOffsetArray], (2 * ((leaf_offset*-1) - 1)) - 1, 2)))), 0),
[varColumnStart])
ELSE 0 END))) END AS hex_Value

FROM @DeletedRecords A
Inner Join sys.allocation_units allocunits On
A.[AllocUnitId]=allocunits.[Allocation_Unit_Id]
INNER JOIN sys.partitions partitions ON (allocunits.type IN (1, 3)
AND partitions.hobt_id = allocunits.container_id) OR (allocunits.type = 2
AND partitions.partition_id = allocunits.container_id)
INNER JOIN sys.system_internals_partition_columns cols ON
cols.partition_id = partitions.partition_id
LEFT OUTER JOIN syscolumns ON syscolumns.id = partitions.object_id AND
syscolumns.colid = cols.partition_column_id
WHERE leaf_offset<0

UNION
/*This part is for fixed data columns*/
SELECT
A.[ID],
Rowlogcontents,
[Transaction ID],
[Slot ID],
NAME ,
cols.leaf_null_bit AS nullbit,
leaf_offset,
ISNULL(syscolumns.length, cols.max_length) AS [length],
cols.system_type_id,
cols.leaf_bit_position AS bitpos,
ISNULL(syscolumns.xprec, cols.precision) AS xprec,
ISNULL(syscolumns.xscale, cols.scale) AS xscale,
SUBSTRING([nullBitMap], cols.leaf_null_bit, 1) AS is_null,
(SELECT TOP 1 ISNULL(SUM(CASE WHEN C.leaf_offset >1 THEN max_length ELSE
0 END),0) FROM
sys.system_internals_partition_columns C WHERE cols.partition_id
=C.partition_id And C.leaf_null_bit<cols.leaf_null_bit)+5 AS [Column
value Size],
syscolumns.length AS [Column Length]

,CASE WHEN SUBSTRING([nullBitMap], cols.leaf_null_bit, 1)=1 THEN NULL
ELSE
SUBSTRING
(
Rowlogcontents,(SELECT TOP 1 ISNULL(SUM(CASE WHEN C.leaf_offset >1 THEN
max_length ELSE 0 END),0) FROM
sys.system_internals_partition_columns C where cols.partition_id
```

```
=C.partition_id And C.leaf_null_bit<cols.leaf_null_bit)+5
,syscolumns.length) END AS hex_Value
FROM @DeletedRecords A
Inner Join sys.allocation_units allocunits ON
A.[AllocUnitId]=allocunits.[Allocation_Unit_Id]
INNER JOIN sys.partitions partitions ON (allocunits.type IN (1, 3)
 AND partitions.hobt_id = allocunits.container_id) OR (allocunits.type =
2 AND partitions.partition_id = allocunits.container_id)
INNER JOIN sys.system_internals_partition_columns cols ON
cols.partition_id = partitions.partition_id
LEFT OUTER JOIN syscolumns ON syscolumns.id = partitions.object_id AND
syscolumns.colid = cols.partition_column_id
WHERE leaf_offset>0 )

--Converting data from Hexvalue to its orgional datatype.
--Implemented datatype conversion mechanism for each datatype
--Select * from sys.columns Where [object_id]=object_id('' +
@SchemaName_n_TableName + '')
--Select * from CTE

INSERT INTO #temp_Data
SELECT
NAME,
CASE
 WHEN system_type_id IN (231, 239) THEN
LTRIM(RTRIM(CONVERT(NVARCHAR(max),hex_Value)))  --NVARCHAR ,NCHAR
 WHEN system_type_id IN (167,175) THEN
LTRIM(RTRIM(CONVERT(VARCHAR(max),REPLACE(hex_Value, 0x00, 0x20))))  --
VARCHAR,CHAR
 WHEN system_type_id = 48 THEN CONVERT(VARCHAR(MAX), CONVERT(TINYINT,
CONVERT(BINARY(1), REVERSE (hex_Value)))) --TINY INTEGER
 WHEN system_type_id = 52 THEN CONVERT(VARCHAR(MAX), CONVERT(SMALLINT,
CONVERT(BINARY(2), REVERSE (hex_Value)))) --SMALL INTEGER
 WHEN system_type_id = 56 THEN CONVERT(VARCHAR(MAX), CONVERT(INT,
CONVERT(BINARY(4), REVERSE(hex_Value)))) -- INTEGER
 WHEN system_type_id = 127 THEN CONVERT(VARCHAR(MAX), CONVERT(BIGINT,
CONVERT(BINARY(8), REVERSE(hex_Value))))-- BIG INTEGER
 WHEN system_type_id = 61 Then
CONVERT(VARCHAR(MAX),CONVERT(DATETIME,CONVERT(VARBINARY(8000),REVERSE
(hex_Value))),100) --DATETIME
 --WHEN system_type_id IN( 40) Then
CONVERT(VARCHAR(MAX),CONVERT(DATE,CONVERT(VARBINARY(8000),(hex_Value))),1
00) --DATE This datatype only works for SQL Server 2008
 WHEN system_type_id =58 Then
CONVERT(VARCHAR(MAX),CONVERT(SMALLDATETIME,CONVERT(VARBINARY(8000),REVERS
E(hex_Value))),100) --SMALL DATETIME
 WHEN system_type_id = 108 THEN CONVERT(VARCHAR(MAX),
CAST(CONVERT(NUMERIC(38,30),
CONVERT(VARBINARY,CONVERT(VARBINARY,xprec)+CONVERT(VARBINARY,xscale))+CON
VERT(VARBINARY(1),0) + hex_Value) as FLOAT)) --- NUMERIC
 WHEN system_type_id In(60,122) THEN
CONVERT(VARCHAR(MAX),Convert(MONEY,Convert(VARBINARY(8000),Reverse(hex_Va
lue))),2) --MONEY,SMALLMONEY
 WHEN system_type_id =106 THEN CONVERT(VARCHAR(MAX),
CAST(CONVERT(Decimal(38,34),
CONVERT(VARBINARY,Convert(VARBINARY,xprec)+CONVERT(VARBINARY,xscale))+CON
VERT(VARBINARY(1),0) + hex_Value) as FLOAT)) --- DECIMAL
```

```
 WHEN system_type_id = 104 THEN CONVERT(VARCHAR(MAX),CONVERT
(BIT,CONVERT(BINARY(1), hex_Value)%2))  -- BIT
 WHEN system_type_id =62 THEN
RTRIM(LTRIM(STR(CONVERT(FLOAT,SIGN(CAST(CONVERT(VARBINARY(8000),Reverse(h
ex_Value)) AS BIGINT)) * (1.0 +
(CAST(CONVERT(VARBINARY(8000),Reverse(hex_Value)) AS BIGINT) &
0x000FFFFFFFFFFFFF) * POWER(CAST(2 AS FLOAT), -52)) * POWER(CAST(2 AS
FLOAT),((CAST(CONVERT(VARBINARY(8000),Reverse(hex_Value)) AS BIGINT) &
0x7ff0000000000000) / EXP(52 * LOG(2))-1023))),53,LEN(hex_Value)))) ---
FLOAT
 When system_type_id =59 THEN
Left(LTRIM(STR(CAST(SIGN(CAST(Convert(VARBINARY(8000),REVERSE(hex_Value))
AS BIGINT))* (1.0 + (CAST(CONVERT(VARBINARY(8000),Reverse(hex_Value)) AS
BIGINT) & 0x007FFFFF) * POWER(CAST(2 AS Real), -23)) * POWER(CAST(2 AS
Real),(((CAST(CONVERT(VARBINARY(8000),Reverse(hex_Value)) AS INT) )&
0x7f800000)/ EXP(23 * LOG(2))-127))AS REAL),23,23)),8) --Real
 WHEN system_type_id In (165,173) THEN (CASE WHEN CHARINDEX(0x,cast('' AS
XML).value('xs:hexBinary(sql:column("hex_Value"))', 'VARBINARY(8000)')) =
0 THEN '0x' ELSE '' END) +cast('' AS
XML).value('xs:hexBinary(sql:column("hex_Value"))', 'varchar(max)') --
BINARY,VARBINARY
 WHEN system_type_id =36 THEN
CONVERT(VARCHAR(MAX),CONVERT(UNIQUEIDENTIFIER,hex_Value)) --
UNIQUEIDENTIFIER
 END AS FieldValue
,[Rowlogcontents]
,[Transaction ID]
,[Slot ID]
,[ID]
FROM CTE ORDER BY nullbit

/*Create Update statement*/
/*Now we have the modified and actual data as well*/
/*We need to create the update statement in case of recovery*/

;With CTE AS (SELECT
(CASE
WHEN system_type_id In (167,175,189) THEN QUOTENAME([Name]) + '=' +
ISNULL(+ '''' + [A].[FieldValue]+ '''','NULL')+ ' ,'+' '
WHEN system_type_id In (231,239) THEN  QUOTENAME([Name]) + '='  +
ISNULL(+ 'N''' +[A].[FieldValue]+ '''','NULL')+ ' ,'+''
WHEN system_type_id In (58,40,61,36) THEN QUOTENAME([Name]) + '='  +
ISNULL(+  ''''+[A].[FieldValue]+ '''','NULL') + '  ,'+' '
WHEN system_type_id In (48,52,56,59,60,62,104,106,108,122,127) THEN
QUOTENAME([Name]) + '='  + ISNULL([A].[FieldValue],'NULL')+ ' ,'+' '
END) as [Field]
,A.[Slot ID]
,A.[Transaction ID] as [Transaction ID]
,'D' AS [Type]
,[A].Rowlogcontents
,[A].[NonID]
FROM #temp_Data AS [A]
INNER JOIN #temp_Data AS [B] ON [A].[FieldName]=[B].[FieldName]
AND [A].[Slot ID]=[B].[Slot ID]
--And [A].[Transaction ID]=[B].[Transaction ID]+1
AND [B].[Transaction ID]=  (SELECT Min(Cast([Transaction ID] as int)) as
[Transaction ID]  FROM #temp_Data AS [C]
```

```
WHERE [A].[Slot ID]=[C].[Slot ID]
GROUP BY [Slot ID])
INNER JOIN sys.columns [D] On  [object_id]=object_id('' +
@SchemaName_n_TableName + '')
AND A.[Fieldname] = D.[name]
WHERE ISNULL([A].[FieldValue],'')<>ISNULL([B].[FieldValue],'')
UNION ALL

SELECT(CASE
WHEN system_type_id In (167,175,189) THEN QUOTENAME([Name]) + '=' +
ISNULL(+ '''' + [A].[FieldValue]+ '''','NULL')+ ' AND '+''
WHEN system_type_id In (231,239) THEN  QUOTENAME([Name]) + '='+ ISNULL(+
'N''' +[A].[FieldValue]+ '''','NULL')+ ' AND '+''
WHEN system_type_id In (58,40,61,36) THEN QUOTENAME([Name]) + '=' +
ISNULL(+  ''''+[A].[FieldValue]+ '''','NULL') + ' AND '+''
WHEN system_type_id In (48,52,56,59,60,62,104,106,108,122,127) THEN
QUOTENAME([Name]) + '='  + ISNULL([A].[FieldValue],'NULL') + ' AND '+''
END) AS [Field]
,A.[Slot ID]
,A.[Transaction ID] AS [Transaction ID]
,'S' AS [Type]
,[A].Rowlogcontents
,[A].[NonID]
FROM #temp_Data AS [A]
INNER JOIN #temp_Data AS [B] ON [A].[FieldName]=[B].[FieldName]
AND [A].[Slot ID]=[B].[Slot ID]
--And [A].[Transaction ID]=[B].[Transaction ID]+1
AND [B].[Transaction ID]=  (SELECT Min(Cast([Transaction ID] as int)) as
[Transaction ID] FROM #temp_Data AS [C]
WHERE [A].[Slot ID]=[C].[Slot ID]
GROUP BY [Slot ID])
INNER JOIN sys.columns [D] ON  [object_id]=object_id('' +
@SchemaName_n_TableName + '')
AND [A].[Fieldname]=D.[name]
WHERE ISNULL([A].[FieldValue],'')=ISNULL([B].[FieldValue],'')
AND A.[Transaction ID] NOT IN (SELECT Min(Cast([Transaction ID] as int))
as [Transaction ID] FROM #temp_Data AS [C]
WHERE [A].[Slot ID]=[C].[Slot ID]
GROUP BY [Slot ID])
)

,CTEUpdateQuery AS (SELECT 'UPDATE ' +  @SchemaName_n_TableName +  ' SET
' + LEFT(
STUFF((SELECT ' ' + ISNULL([Field],'')+ ' ' FROM CTE B
WHERE A.[Slot ID]=B.[Slot ID] AND A.[Transaction ID]=B.[Transaction ID]
And B.[Type]='D' FOR XML PATH('') ),1,1,''),

LEN(STUFF((SELECT ' ' +ISNULL([Field],'')+ ' ' FROM CTE B
WHERE A.[Slot ID]=B.[Slot ID] AND A.[Transaction ID]=B.[Transaction ID]
And B.[Type]='D' FOR XML PATH('') ),1,1,'') )-2)

+ '  WHERE  ' +

LEFT(STUFF((SELECT ' ' +ISNULL([Field],'')+ ' ' FROM CTE C
WHERE A.[Slot ID]=C.[Slot ID] AND A.[Transaction ID]=C.[Transaction ID]
And C.[Type]='S' FOR XML PATH('') ),1,1,'') ,
```

```
LEN(STUFF((SELECT ' ' +ISNULL([Field],'')+ ' ' FROM CTE C
WHERE A.[Slot ID]=C.[Slot ID] AND A.[Transaction ID]=C.[Transaction ID]
And C.[Type]='S' FOR XML PATH('') ),1,1,''))-4)
AS [Update Statement],
[Slot ID]
,[Transaction ID]
,Rowlogcontents
,[A].[NonID]
FROM CTE A
GROUP BY [Slot ID]
,[Transaction ID]
,Rowlogcontents
,[A].[NonID] )

INSERT INTO #temp_Data
SELECT 'Update Statement',ISNULL([Update
Statement],''),[Rowlogcontents],[Transaction ID],[Slot ID],[NonID] FROM
CTEUpdateQuery

/*************************/
--Create the column name in the same order to do pivot table.
DECLARE @FieldName VARCHAR(max)
SET @FieldName = STUFF(
(
SELECT ',' + CAST(QUOTENAME([Name]) AS VARCHAR(MAX)) FROM syscolumns
WHERE id=object_id('' + @SchemaName_n_TableName + '')

FOR XML PATH('')
), 1, 1, '')

--Finally did pivot table and got the data back in the same format.
--The [Update Statement] column will give you the query that you can
execute in case of recovery.
SET @sql = 'SELECT ' + @FieldName  + ',[Update Statement] FROM #temp_Data
PIVOT (Min([FieldValue]) FOR FieldName IN (' + @FieldName  + ',[Update
Statement])) AS pvt
Where [Transaction ID] NOT In (Select Min(Cast([Transaction ID] as int))
as [Transaction ID] from #temp_Data
Group By [Slot ID]) ORDER BY Convert(int,[Slot
ID]),Convert(int,[Transaction ID])'
Print @sql
EXEC sp_executesql @sql
```

۹-۳ پیوست ج

```
CREATE PROCEDURE Recover_Dropped_Objects_Detail_Proc
@Date_From DATETIME='1900/01/01',
@Date_To DATETIME ='9999/12/31'
AS

;WITH CTE AS (Select B.name AS [Schema Name]
,REPLACE(SUBSTRING(A.[RowLog Contents 0],14,LEN(A.[RowLog Contents
0])),0x00,0x) AS [Object Name]
,[Transaction ID]
,A.[RowLog Contents 0]
FROM fn_dblog(NULL,NULL) A
LEFT JOIN sys.schemas B
```

```
ON CONVERT(INT,SUBSTRING([RowLog Contents 0],2,2))= B.schema_id
WHERE A.[AllocUnitName] ='sys.sysschobjs.nc1'AND
A.[Transaction ID] IN (
SELECT DISTINCT [TRANSACTION ID] FROM  sys.fn_dblog(NULL, NULL)
WHERE Context IN ('LCX_NULL') AND Operation IN ('LOP_BEGIN_XACT')
AND [Transaction Name] LIKE '%DROP%'
AND CONVERT(NVARCHAR(11),[Begin Time]) BETWEEN @Date_From AND @Date_To))

SELECT
[Schema Name]
,[Object Name]
,B.[Begin Time]  AS [Dropped Date & Time]
,C.[name] AS [Dropped By User Name]
FROM CTE A
INNER JOIN fn_dblog(NULL,NULL) B
ON A.[Transaction ID] =B.[Transaction ID]
AND Context IN ('LCX_NULL') AND Operation IN ('LOP_BEGIN_XACT')
AND [Transaction Name]LIKE '%DROP%'
INNER JOIN sys.sysusers  C ON B.[Transaction SID]=C.[Sid]
GO


--EXEC Recover_Dropped_Objects_Detail_Proc
```