The Intersection of Compliance, Databases, and IT Operations

Ben Lenard
DePaul University
Argonne National Laboratory
Chicago, IL, USA
blenard@anl.gov

Alexander Rasin
DePaul University
Chicago, IL, USA
arasin@cdm.depaul.edu

Nick Scope
DePaul University
Chicago, IL, USA
nscope52884@gmail.com

Thamer Al Johani
DePaul University
Chicago, IL, USA
taljoha2@depaul.edu

Abstract

Most organizations rely on relational database(s) for their day-to-day business functions. Data management policies fall under the umbrella of IT Operations, dictated by a combination of internal organizational policies and government regulations. Many privacy laws (such as Europe's General Data Protection Regulation and California's Consumer Privacy Act) establish policy requirements for organizations, requiring the preservation or purging of certain customer data across their systems. Organization disaster recovery policies also mandate backup policies to prevent data loss. Thus, the data in these databases are subject to a range of policies, including data retention and data purging rules, which may come into conflict with the need for regular backups.

In this paper, we discuss the trade-offs between different compliance mechanisms to maintain IT Operational policies. We consider the practical availability of data in an active relational database and in a backup, including: 1) supporting data privacy rules with respect to preserving or purging customer data, and 2) the application performance impact caused by the database policy implementation. We first discuss the state of data privacy compliance in database systems. We then look at enforcement of common IT operational policies with regard to database backups. We consider different implementations used to enforce privacy rule compliance combined with a detailed discussion for how these approaches impact the performance of a database at different phases. We demonstrate that naive compliance implementations will incur a prohibitively high cost and impose onerous restrictions on backup and restore process, but will not affect daily user query transaction cost. However, we also show that other solutions can achieve a far lower backup and restore costs at a price of a small (<5%) overhead to non-SELECT queries.

CCS Concepts

• **Applied computing** → IT governance; *Enterprise data management*; • **Security and privacy** → **Database and storage security**.

SSDBM 2024, July 10–12, 2024, Rennes, France

ACM ISBN 979-8-4007-1020-9/24/07
https://doi.org/10.1145/3676288.3676297

Keywords

Relational Database, Compliance, Operational Procedures, Backup and Recovery

ACM Reference Format:
Ben Lenard, Alexander Rasin, Nick Scope, and Thamer Al Johani. 2024. The Intersection of Compliance, Databases, and IT Operations. In *36th International Conference on Scientific and Statistical Database Management (SSDBM 2024), July 10–12, 2024, Rennes, France.* ACM, New York, NY, USA, 10 pages. https://doi.org/10.1145/3676288.3676297

1 Introduction

Most modern businesses are dependent on relational database management systems (RDBMS) to operate and exist as a business; very few businesses can survive without their data as the data defines them. In fact, this data is often used to build their competitive advantage, or what makes the business unique. Since this data is key to business success, one of the responsibilities of the IT department is ensuring that this data is safe from destructive events, such as hardware failures or an external disaster. A disaster could be a natural event such as a hurricane or an earthquake; however, a disaster could also be an upgrade gone wrong, failed hardware, rogue employee, or human error. Planning for recovery effectively is crucial, especially in light of the unique difficulties caused by regional differences in natural catastrophes. For example, with Hurricane Sandy, we saw the devastation in data centers too close to the coast line; many of these data centers became flooded despite having redundant equipment [18, 23]. Thus, it may be no longer safe to have a "remote" disaster recovery site in the same region. While each region within the United States may have its own set of natural disasters, Sandy caused people to reevaluate disaster recovery and business continuity. In fact, some telecommunications companies are trying to forecast what climate change means for their infrastructure in thirty years [2].

One of the key tasks performed by the IT Operations group is to develop a Disaster Recovery (DR) and Business Continuity (BC) Plan. DR and BC plan has to account for both natural disasters (e.g., an earthquake) as well as malicious incidents such as ransomware attacks. These plans often use backups in a cold, warm, or hot secondary location and are essential for the organization should something go awry; a University of Texas study by Christens and Schkade [11] found that 94% of business who suffer a major data

loss close their doors; another study found 60% close within six months [42].

To accommodate the demand for customer data privacy, new laws and policies have been introduced to require the retention (based on necessity for data) and purging of records when the data is no longer required for business reasons. If fact, in addition to California Consumer Privacy Act (CCPA), there are 31 [4] other states that have recently enacted or updated laws dictating the management and storage of data. That being said, if an organization had a data governance strategy, the retention and purging implementation would likely fall on the data steward to develop these policies and communicate to the data custodian to implement them, if the necessary features were not already built into the application. A data steward is an individual who is responsible for designing, implementing and enforcing data management policies and procedures [28]. In a less complex corporate environment (i.e., smaller organizations), the business and IT Operations would work together to develop and implement these rules so that the business remains compliant. For example, when IT Operations delete data from an RDBMS, they expect the data to be destroyed since the SQL statement does not return deleted data. In practice, however, this deleted data still remains on disk (i.e., it is not actually gone and can still be recovered). In fact, deleted data in an RDBMS has a surprisingly long lifetime [21, 39].

A related key responsibility of IT Operations is database performance, which greatly impacts general application performance. A database administrator (DBA) continuously monitors and tunes the database performance, so that the application maintains its performance level. Within a structured environment this could be guided by a formal Service Level Agreement (SLA); in an application that faces users, one has a few seconds before a user navigates away because the user is unhappy [14]. In an internal application, employees are the users who will instead file (dreaded) support tickets that say "It's slow" [26].

This paper evaluates several viable options for enforcing compliance for retention of data and purging of data from both active and backup tables. To our knowledge, there have not been any previous attempts to measure the overhead of data storage compliance support in a DBMS. Since relational DBMS behavior is similar across all major vendors (commercial and open source), our analysis generalizes to any major relational database. Our major contributions in this paper include:

- Review and compare all plausible‘ strategies available to enforce compliance (both retention and purging) in a typical relational database
- Evaluate the runtime overhead imposed by different frameworks with purging and retention policies enforcement at a different scale
- Compare the overhead of different existing strategies for purging data in (possibly offline) backups at a different data scale

2 Background and Related Work

Business Record: Organizational rules and requirements for data management are defined in units of business records. United States federal law refers to a business record broadly as as any "memorandum, writing, entry, print, representation or combination thereof, of any act, transaction, occurrence, or event [that is] kept or recorded [by any] business institution, member of a profession or calling, or any department or agency of government [...] in the regular course of business or activity" [36]. In other words, business records describe any interaction or transaction resulting in new data. In a database, a business record may span many combinations of rows across multiple tables (e.g., a purchase order consisting of a buyer, a product, and the purchase transaction from three different tables). Compliance solutions (e.g., Ataullah et al. [9]) define business records using Select-Project-Join SQL syntax.

Retention: Retention policies mandate the preservation of data by protecting it from deletion and optionally maintaining a comprehensive update history. Retention may be defined for a certain period or an indefinite amount of time. Some retention requirements (e.g., the Health Insurance Portability and Accountability Act) may require a complete historical log of all business record updates (e.g., current address and full history of address changes for a patient) [10].

Purging: Purging is the permanent and irreversible destruction of data in a business record [17]. A business record purge can be accomplished by physically destroying the device which stored the data, digitally erasing all data from the device, or encrypting data and erasing the corresponding decryption key (although the ciphertext still exists, destroying the decryption key makes it irrecoverable in practice). For example, if a file is deleted in a file system, but can still be recovered from storage using forensic tools, it does not qualify as purged [17]. Similarly, records deleted in a database can be recovered through forensic analysis [38] and queried from all physical storage [40]; such records and other partial data elements [41] can be retained indefinitely within database backups [21].

Business Impact Analysis: A Business Impact Analysis identifies business processes, potential impacts to business success, as well the likelihood of them occurring [34]. For example, a delivery company might look at diesel fuel shortages, union strikes, and vehicle part shortages, considering the impact and the likelihood of these events occurring. The impact considerations would also include the path to recovery if such an event occurred.

Business Continuity (BC) and Disaster Recovery (DR):. BC and DR are critical components of organizational resilience, aimed at ensuring operational continuity during adverse events, regardless of the cause [24]. It extends beyond IT to encompass crisis management, employee safety, and alternative work arrangements. A comprehensive BC and DR approach involves meticulous planning, including Business Impact Analysis, risk analysis, and the creation of plans, tests, exercises, and training. Planning documents serve as a vital resource, containing employee contacts, emergency details, vendor information, testing procedures, equipment lists, and technical diagrams.

Legal hold: This obligation is known commonly as a litigation hold. Between 2003 and 2004 in New York, the case of Zubulake v. UBS Warburg elaborated on the application of a litigation hold

to electronically stored information [12]. As a result of the ruling, organizations must retain and preserve relevant data even if they merely anticipate a lawsuit in the future.

Service Level Agreement: A Service Level Agreement (SLA) defines the service expected from a vendor or application. Metrics are used to define and quantify the delivered service [8]. For example, availability and up time can often be subject to an SLA, defining how much the service offered can be down. If a vendor or an application does not meet their SLA, a compensation might be warranted, depending on the agreement.

Academic Research: Lenard et al. [21] showed that deleted (yet forensically recoverable) data will be incorporated into database backups. When a database deletes a row, the row is soft deleted, meaning a delete flag is set, but the row exists until the table is reorganized and the row is overwritten. Database software performs backups at the block level, copying such deleted data into the backups unless the table is reorganized prior to the backup. Even then, a window of opportunity exists where deletes could occur and make their way into a backup. Ultimately, it is up to the organization to determine the data lifecycle phases at which noncompliance with data handling may occur [30]. Li et al. [22] designed a system that supports policy-based object erasure using cryptographic erasure (i.e., encrypt the data and then destroy the key to prevent data access).

Ataullah et al. [9] presented a retention approach in relational databases that blocks queries (using triggers) whose execution would result in a non-compliant database state. They pioneered using Select-Project-Join (SPJ) queries to define business records and policies. When a `DELETE` or `UPDATE` query came into conflict with a retention policy, the framework would prevent the query from executing.

Scope et al. [29] developed a compliance system that implements both retention and purging mechanisms in a database using triggers and SPJ definitions (similar to Ataullah et al. [9]). Retention is implemented by transparently archiving the data as its status changes [32]; because the user deleted the data, it is removed from active database tables yet is retained in the archive for compliance purposes. Purging is implemented through cryptographic erasure by mirroring regular tables in so called "shadow tables" which encrypt all values subject to purging in the future [33]. Both archive and shadow tables are maintained with triggers as `INSERT`s and `UPDATE`s are executed; shadow tables replace original tables in database backup to ensure purging capability. Cryptographic erasure mechanic also addresses the problem of forensically recovered content that may be remain in backups [21], as long as the data is subject to a purging policy.

Industry Tools: Google Cloud offers retention protection policies that can be placed on objects to prohibit premature erasure. Furthermore, Google's Data lifecycle management functionality allows the storage administrators to set real-time policies that automatically move object to Google's coldline store (i.e., archive) or delete objects based on predefined rules. Currently, Google does not offer the ability to automatically archive objects instead of deletion. Amazon's AWS [37] offers many of the same features as Google for potential compliance support. It offers the ability to encrypt objects, set life cycle rules for prohibiting premature erasure, migrate objects to lower cost archive storage (AWS Glacier), and automatically deleting objects based on a time criteria. Overall, many of these features are common across major cloud storage providers.

Oracle's Flashback Data Achieve (Total Recall) allows users to recall the history of a table at the tuple level [27] (thus it does not directly fulfill organizational needs for a business record level policy across tables). Archiving automatically provides a history of business records whenever a value of the business record is `UPDATE`d. While Total Recall provides retention capabilities, it does not provide a solution for expunging data from backups.

None of the above industry solutions support deletion at a finer granularity than the file level. This paper specifically benchmarks the performance of supporting retention and purging at the tuple level, and thus, we do not offer direct comparison to the existing industry tools. A comparison to industry tools would be reduced to providing a comparison of file erasure speed, which is contingent on the server performance of the vendor and the target file size.

3 Motivation

A database administrator who implements backup policies must refer to the organization's disaster recovery and business continuity requirements. There is an entire spectrum of backup methodologies, ranging from full backups (e.g., weekly or monthly), incremental backups, or differential backups in between (see [21] for an overview). Backups are an operational requirement that has evolved to ensure data safety; 94% of businesses that suffered a catastrophic data loss shut their doors within two years [11]. Regardless of specific regulatory policy for disaster recovery and business continuity plans, a responsible IT organisation, private or public, must have backups to recover from a disaster regardless of the disaster cause or how predictable it may be. Additional backups must often be kept in an external physical location so that data can be recovered if something happens to the data center itself (regardless of whether the data center is cloud-based or not) [19].

Furthermore, regardless of the backup method, we know that deleted data (i.e., deleted by user but still forensically recoverable [39]) will trickle into backups unless data tables are explicitly rebuilt prior to the backup creation [21]. For a small database, rebuilding data structures before backup may be practical. However, for a larger database, the reorganization process would take time, CPU time, as well as IO per second (IOPS). While an online reorganization could happen in the background, and one could throttle the process, it is still consuming CPU and IOPS, and that could, and often does, impact application performance.

Application Performance also falls under the umbrella of IT Operations. While some applications may have an audience that is not sensitive to a slowdown in response time, many applications have a set window before the user gets impatient and moves on (e.g., leaving the company website for a competitor), or starts complaining [13, 16]. Consider examples such as a financial trading application, a global logistics company, or a credit card network. Most applications have a standard service level agreement (SLA) that is measured in seconds. For example, when one physically uses a credit card, the authorization needs to come back in seconds. Behind the scenes, the vendor had to transmit the credit card info to

their processor, who then had to contact the network, which then contacts the underwriting bank for approval before responding back with an authorization code [3].

Because SLA are a requirement that may be costly to violate, there are suites of tools specifically centered around application performance management, such as IBM APM [6] or IBM Db2 Data Management Console [7]. These can send alerts about queries exceeding a specified performance threshold (to monitor and ensure the SLAs are not violated). In the credit card example, there are multiple steps in the credit card process, and there are multiple different systems implementing these steps. The application performance monitoring tools ensure that the response times meet their SLAs.

In this paper, we consider the impact of compliance support on database performance in different stages of user query performance and backup process, thus identifying which processes may be significantly affected by compliance enforcement overheads. Database tuning is a significant factor in the overall responsibilities of improving application performance tasks of IT operations.

There are a variety of reasons why a company would retain business records; the reasons could range from a legal hold, to a regulatory rule from some government legislation, or simply to protect it from accidental deletion. Regardless, retention requirements are a regular focal point of data lifecycle planning [31], particularly in finance and healthcare industries. For example, if one were to receive a legal hold for a customer's data, IT Operations is responsible for ensuring that the application has the appropriate tools to ensure the customer's data was retained to comply with legal requirements. Furthermore, this would require the capability to first correctly identify all records and values relevant to the issued legal hold. On the other hand, if one were to delete data after an event (e.g., due to a GDPR "right to be forgotten" request), IT Operations is responsible for ensuring that this deleted data does not contaminate the backups. If purged customer data remains in backups, the data has not been fully purged. While Oracle provides a database specific function for retention, namely Total Recall, not every RDBMS or application runs on Oracle, nor does it support purging.

As CPRA and other laws require the purging of customer data, this is an aspect of compliance that must be harmonized with IT operations. For example, Kentucky law *50-7a03, Destruction of consumer information* says "Unless otherwise required by federal law or regulation, a person or business shall take reasonable steps to destroy or arrange for the destruction of a customer's records within its custody or control containing personal information which is no longer to be retained by the person or business by shredding, erasing or otherwise modifying the personal information in the records to make it unreadable or undecipherable through any means." As discussed earlier, deleted data survives in database structures, propagating to the block based backups. It is the responsibility of the data curators to be aware of the applicable data purging requirements and of the technical details that may result in non-compliance. Many laws and policies state that the businesses are not permitted to hold data indefinitely and that companies must purge data after a given period of time. While database administrator might be able to delete data pertaining to the specific customers under a law's requirements, they have no reliable mechanism to measure compliance of their storage and backups. New York's SHIELD Act requires that businesses "develop, implement and maintain reasonable safeguards to protect the security, confidentiality and integrity of the private information including, but not limited to, disposal of data." In sum, there are many jurisdiction-specific and policy-specific reasons why data must eventually be disposed of.

Given that an organization may need to retain database backups for the survival of the business, while maintaining regulatory compliance for deleted and archived data and minimizing active storage overhead to maximize performance, we break down the performance considerations across possible approaches. Any compliance implementation must find an approach that satisfies a multitude of stakeholders within an organization.

4 Evaluation

In this section, we evaluate frameworks that address different IT Operations that pertain to data storage compliance. Specifically, we consider Ataullah et al. [9] retention framework, Scope et al. [29] retention and purging framework, and two simple custom strategies for manually purging data from backups by editing the backups to remove purged data. We compare these approaches for both data retention and purging purposes, measuring the overhead (both for storage and performance) as well as for ease of IT deployment. Table 1 summarizes all of the approaches and their applicability to retention and purging. Ataullah et al. and Scope et al. approaches are described in Section 2; the two custom reference approaches are discussed next.

Restore, `DELETE`*, and Backup:* Using a restore, `DELETE`, and backup (RDB) process requires connecting to the database backup and modifying it directly to accomodate data purging. The downside of that approach is the requirement that all backups remain accessible at all times. Thus, storing the backup on a physically disconnected storage medium would prohibit this process or impose a further requirement that backups be recalled from offsite storage (an action which would be required at every designated purging period bucket). Many organizations use delta and incremental backups in addition to periodic full backups. Both of these approaches (delta and incremental) capture changes to the database since the last full backup. Editing a backup to purge data would further void all delta and incremental backups associated with the full backup that was modified. In our evaluation, we assume that all backups are accessible and only full backups are used (which is typically not the case in practice).

We also assume that some steps were taken to optimize the RDB purging process, such as removing indexes during the restore (to avoid index update costs). While we performed our evaluation for one database, this would be a more expensive proposition for an enterprise where they could have hundreds to thousands of different databases (and corresponding backups). Furthermore, these approaches get more expensive when the backups are stored offsite for business continuity and disaster recovery purposes. In order to edit a backup, one would have to recall the stored backup, load the backup, delete the data, recreated the backup, and then ship the storage to its designated offsite storage facility. While automated processes could streamline this process, it would still require significant additional computing resources of the organization. For

	Retention Support	Purging Support	Section
Ataullah et al. [9]	Yes	-	4.1
Scope et al. [29]	Yes	Yes	4.1, 4.2
Restore, DELETE, and Backup (RDB)	-	Yes	4.2
Partitioning Backups	-	Yes	4.2

Table 1: Approaches evaluated in this paper

backup implementation, we used `pg_dump` which provides a consistent backup of the database; `pg_dump` is a utility provided with Postgres for exporting a Postgres database.

Partitioning Backups: This approach exports the business records into individual files partitioned by purging date. Once the business records are bucketed by the purge date range, they can be removed by deleting the file that corresponds to the relevant partition of the table. The to-be-purged data is clustered together by this partitioning process. Once all business records have been exported and removed, the remaining data in the database is backed up using regular backup process. As with RDB approach, in order to purge the files to maintain compliance, each backup file requiring purging must be physically accessible (i.e., online) at the time of its corresponding purge period.

Our original intent was to implement table partitioning using views or material views. However, `pg_dump` command does not export views. Thus, we used temporary tables as a partitioning mechanism, with the directory option as the target of `pg_dump` so that each table is exported into its own file. Since every table is a file, this implementation breaks tables into smaller temporary tables so that individual purge date ranges can be deleted. During the creation of temporary tables, the application must be stopped so that a consistent database backup can be created. When restoring from this backup, the removed chunk of the table would be skipped when loading into the target table. This process will present some challenges in practice as many larger database systems do not allow database outages for even a short period of time.

Dataset: In our experiments, we use the TPC-H benchmark [35], for both examples and for our experimental evaluation. TPC-H is a recognized industry benchmark, preferable to a synthetic schema or a complex enterprise application, such as SAP or PeopleSoft. For the purposes of our analysis, we make the following changes to TPC-H:

(1) Removed referential integrity constraints between:
   - `LINEITEM` and `PARTSUPP` tables
   - `CUSTOMER` and `NATION` tables
(2) Added the following columns to the tables `CUSTOMER`, `ORDERS`, and `LINEITEM`:
   - `GROUP`: Used to set policies on controlled subsets of the table data
   - `RETAIN_DATE`: As the criteria for the retention duration
   - `PURGE_DATE`: As the criteria for when the business record must be destroyed

The first change was necessary to allow us to set policies to fit our parameters for testing; specifically, we wanted to test policies covering one, two, and three tables without causing violations of referential integrity. The second change was needed to allow us to bucket the groups for different data coverage, while the dates allowed us to balance the distribution of data covered by retention and purging criteria.

We used a server with dual Intel Xeon E5645, each with 6 physical cores and Hyper Threading enabled, 64GB of RAM, and an SSD drive. The server was running CentOS 8 Stream x86_64 with Kernel Virtual Machine [20] (KVM) as the hypervisor software. We used two Virtual Machines (VMs) to carry out the experiments; since a majority of database interactions operate in a client-server model, we deployed two independent VMs to represent client and server. Both VMs were built with CentOS 8 Stream x86_64, Postgres 14.5, 1 x vNIC and a 25GB QEMU copy-on-write [5] (QCOW2) file on an SSD. The client VM has 4GB of RAM and 4 vCPUs and the server VM was allocated 8GB of RAM and 4 vCPUs. The Postgres Buffer Pool was set to 4GB. The QCOW2 file was partitioned into: 350MB /boot, 2GB swap space, with the remaining storage used for the / partition, using standard partitioning and ext4 file system. Only these two VMs were running on the hypervisor to minimize runtime fluctuations. For Scale 10 TPC-H experiments, we had to expand the QCOW2 file to 100GB and then expand the filesystem with native utilities.

We ran a combination of `DELETE`, `UPDATE`, and `SELECT` queries (using the standard TPC-H `SELECT` queries 1-16) across different implementations to measure the overheads imposed by various solutions. We evaluated retention blocking and cryptographic erasure frameworks by adjusting the data size, the number of active policies, the percent of tuples in the tables covered by the policies, the number of tables in the policies, and the scale of the database. The parameters summarized in Table 2 were tested on a system which had no compliance framework (i.e., the baseline times), retention protections that automatically archived data, retention protections that blocked queries that would result in non-compliance, and a simultaneous retention and purging framework that utilizes archiving and encryption for cryptographic erasure.

Variable	**Inputs**
Tables Covered by Policy	1, 2, 3
Coverage	0.1 - 1
Policies	1 - 20
Scale (i.e., Total GB of Data)	1, 3, 10

Table 2: Input Variables Tested

For each workload tested, we ran 9,400 `SELECT` queries, 5,000 `UPDATE` queries targeting a table with an active policy, 5,000 `UPDATE` queries targeting other tables in the database, 5,000 `DELETE` queries targeting a table with an active policy, and 5,000 `DELETE` queries

targeting other tables in the database. These queries were shuffled and the query times were measured per-query using real time measure.

For our statistical hypothesis, the default (i.e., null hypothesis) assumption is denoted as H_0. When we found the probability of the null hypothesis to be statistically unlikely (in our case using a p-value of < 0.05), we conclude the alternative hypothesis (H_a) to be true.

Across our analyses, we consider the overhead on a database that approximates real-world transactions. Hsu et al. [15] previously found that typical data warehouse query workloads are, on average, 90% `SELECT`s, 7% `UPDATE`s, and 3% `DELETE`s. During the evaluation of the overhead, each combination of our tested variables is compared to the runtime of the same query without any retention or purging functionality. We then calculate the average weighted overhead for `SELECT`s, `UPDATE`s, and `DELETE`s using the overhead of each query. For the overall workload overhead, we only use the overhead of `UPDATE`s and `DELETE`s targeting tables that had an active policy. This pessimistic assumption provides the highest potential overhead. Using O to denote overhead, Eq. 1 denotes our overhead formula for a query workload:

$$O_{workload} = (O_{select} * 0.90) + (O_{update} * 0.07) + (O_{delete} * 0.03) \quad (1)$$

When discussing workload overhead, we calculate the performance of a given approach compared to a non-compliant database using Eq. 2:

$$O = \frac{performance_{compliant} - performance_{non-compliant}}{performance_{non-compliant}} \quad (2)$$

Therefore, an $O > 0$ translates to an overhead added to the performance of a workload when compared to a non-compliant system. For example, a $O = 0.10$ would roughly translate to a 10% performance overhead.

SELECT Overhead: Because trigger-based solutions (Ataullah et al. [9] and Scope et al. [29]) do not execute when running a `SELECT` query, we first verify the expected overhead in `SELECT` queries. We tested by comparing the simultaneous retention and purging approach against a database without any compliance framework using 9,400 `SELECT`s across the workload combinations. This analysis was performed using both indexed and non-indexed workloads. Our hypotheses were:

$$H_0 : O_{select} = 0$$
$$H_a : O_{select} > 0$$

Using a one-sample t-test (which assumes normality) and a Wilcoxon signed-rank test (a non-parametric test), we fail to reject H_0 of the overhead of `SELECT` queries being equal to 0 (both returned a p-value of 1.0). Thus, when calculating the overhead of framework we assume `SELECT` overhead of 0.

Indexing. To evaluate whether or not indexing additional columns would increase the relative performance overhead of a compliance framework, we ran workload combinations using a subset of our variables with and without indexed columns in the database used in a retention and purging compliance framework. Our hypotheses are as follows:

$$H_0 : O_{indexed\ workloads} = O_{non-indexed\ workloads}$$
$$H_a : O_{indexed\ workloads} \neq O_{non-indexed\ workloads}$$

These two distributions are compared using a two-sample t-test; this test fails to reject H_0 of the indexing overhead being equal to the non-indexing overhead (with a p-value of 0.059). In subsequent analyses, we default to only using indexed workloads. Note that query performance with and without indexing has changed; however, the relative query overhead remained the same.

4.1 Retention: Archiving or Blocking

In this analysis, we compare the retention archiving framework (Scope et al. [29]) to a retention blocking framework (Ataullah et al. [9]). We ran workloads using the variable combinations outlined in Table 2 with indexing enabled. The average workload overhead with retention blocking was 0.00 while the average workload overhead with the retention archiving framework enabled was 0.03. These averages are analyzed using a two-sample t-test with the following hypotheses:

$$H_0 : O_{retention\ blocking} = O_{retention\ archiving}$$
$$H_a : O_{retention\ blocking} \neq O_{retention\ archiving}$$

This test returns a p-value of < 0.001, which allows us to reject H_0 and conclude that a retention archiving framework compared to a retention blocking framework come with a statistically significant difference in the workload overhead. Overall, the retention archiving framework has an average overhead of 3% when analyzed using our query workload.

4.2 Purging: Backup Solution Comparison

In our analysis of backup purging approaches, we divide the analysis into three steps: 1) backup creation, 2) purging, and 3) backup restore. Because the database itself does not support purging in backups, we cannot use it as a baseline. Thus, we use the cryptographic erasure framework (Scope et al. [29]) as the baseline time; i.e., all times are reported relative to the cryptographic framework performance.

We used the TPC-H modified schema with purging policies that cover all columns for the `lineitem` table. We begin our analysis creating the backups with an as-of-date of 12/31/2022 with databases that contain records that must be purged over the next six (6) months. Next, we simulated database changes one, two, and three months ahead to evaluate the cost of implementing different compliance solutions. In these simulated futures, the backups contain both data that has already expired and as well as data that is still retained. Policy expiration ranges were bucketed by month for this analysis (i.e., the same encryption key or partition combined all records requiring purging into the same month granularity). In practice, organizations may choose to implement a more narrow or broad date time-periods depending on organizational need and policy requirements.

With respect to the partitioning approach, our comparison simplified this process by having a single table where all columns were included under the purge policy. In practice, business records may span multiple tables with the additional complexity of potentially only a subset of the table columns defined under the policy. If one were to only include subsets of columns from different tables with partitioning, the restore process would require a custom Extract Transform and Load (ETL) process to re-normalize the denormalized data. This would require either A) reconstructing tables from a series of temporary tables or B) using a combination of `INSERT`s and `UPDATE`s.

Table 3 provides the sizes associated with the main components of a cryptographic erasure approach. This approach can be divided into three main components: 1) the active tables, 2) the table used to manage and store the encryption keys and corresponding purge dates, and 3) the shadow tables used to store the encrypted copies of data. The size of the encrypted data can fluctuate greatly depending on the encryption key used. When an encryption key is deleted, the corresponding encrypted data still takes up space on disk, even though it is no longer recoverable.

Backup	Size (MB)
Active Tables	1,419.19
Encryption Key Table	55.73
`lineItem_shadow` Tables	4745.93

Table 3: Cryptographic Erasure Backup Sizes

Table 4 summarizes the corresponding sizes of a database using the RDB approach. This implementation requires a full restore of the database to purge the data, but this approach enables us to remove and reuse the purged storage on disk. Depending on the pages and whether or not a database defragmentation was executed (i.e., the tables were rebuilt), these numbers can also fluctuate (see the analysis by Lenard et al. [21]) depending on database engine and query workload.

Backup	Size (MB)
Starting Backup	1,419.19
Backup After 1 Month of Purging	1,403.45
Backup After 2 Months of Purging	1,362.99
Backup After 3 Months of Purging	1,289.08

Table 4: Restore, `DELETE`, Backup Database Backup Sizes

The third approach of partitioning the records by purge date resulted in file sizes shown in Table 5. In our analysis, we broke up the purge dates into six months, with data not subject to purge policy rules backed up together in a single file. As with the RDB approach, once a file is deleted, the system can mark the storage space as free for other purposes (which is not the case with a cryptographic erasure approach).

Performance Overhead: To analyze the performance cost of maintaining compliant backups, we consider the cost of creating backups, purging backups to maintain compliance, and restoring data from a compliant backup. Neither the partitioning approach nor the RDB solution would require any changes to the day-to-day use of a database; thus, they do not incur any performance overhead with standard `SELECT`, `DELETE`, and `UPDATE` queries in a database. On the other hand, the cryptographic purging framework does incur a day-to-day query performance overhead as discussed previously.

Backup	Size (MB)
Backup Size (minus Records Subject to Purging)	390.98
Average Monthly Purge Size	171.37

Table 5: Partitioning Backup Sizes

For each of the analyses, we first calculate the average time the baseline framework takes to execute the given step (e.g., creating a backup), denoted as μ. We then compare that framework's performance to the performance of other approaches for the same task (denoted as α) using the formula in Eq. 3:

$$\text{relative performance} = \frac{(\mu - \alpha)}{\mu} \tag{3}$$

The relative performance scale is in $(-\inf : 1]$ range. The performance that is the same as the baseline corresponds to a zero. Thus, a negative value ($\alpha > \mu$) indicates a comparatively slower performance compared to the baseline framework while a positive result ($\alpha < \mu$) indicates a comparatively faster performance.

Creating Backups: When creating a backup using the RDB implementation, initial backup creation does not add any steps to the default process. Thus, RDB adds almost no overhead to that phase; compared to the cryptographic erasure framework, this approach has an average relative performance of 0.92, indicating a much faster performance.

Using partitioning to create separate files for purging requires non-trivial changes to the backup process. Because Postgres does not support exporting views as separate backup files, we first create temporary tables to store the partitioned backups. These temporary tables are then exported with the other tables not subject to the purge policy requirements. The backup creation process for the partitioning approach was found to have an average relative performance of 0.49 – also significantly faster than the baseline but slower than RDB's.

Overall, both of these approaches had a statistically significant different relative performance compared to the cryptographic erasure framework. However, we believe that the lower cost of RDB and partitioning approaches is not worth the downtime that they impose on the database. While the `pg_dump` command ensures a consistent backup, it does not support export of views. Thus, one must stop the application or take alternative steps to ensure the database is in a consistent state before creating the temporary tables and running the steps necessary to partition business records for exporting into separate files. Although the cryptographic erasure framework approach takes longer due to the additional storage in a backup requiring extra storage for the encrypted values, it does not require any application downtime.

Purging Data to Maintain Compliance: In order to purge data with the RDB approach, one must restore the entire backup in order

to purge the expired records. Adding to the expected cost, this must be done for every backup that contains relevant data; if an organization were to have multiple backups, this processing time would be multiplied by the number of backups that contain data subject to purging. When comparing the cryptographic erasure framework to a single backup using the RDB process, the RDB process was found to have a relative performance of -98.53 (indicating an extremely high relative cost).

With the partitioning-based approach, the only step required to purge data is deleting the files with expired data at the OS level (i.e., no restore is required). This approach showed an average relative performance of 0.45 indicating a faster performance. As with the RDB process, if one were to have multiple backups subject to a purge, this would result in a slower performance when compared to the baseline cryptographic erasure framework. Additionally, if we were to bucket the purging period of the records in smaller buckets (e.g., by day or by week instead of by month), this would require deleting more files leading to a slower performance.

Both of the relative performance comparisons were found to exhibit a statistically significant difference when compared the baseline cryptographic erasure framework. Another important consideration of both RDB and partitioning approaches is the requirement of the backups/partitioned files being accessible in order to execute the purging steps. The cryptographic erasure framework does not require the encrypted data in backup(s) to be accessible, although it does require access to the encryption key table.

Restoring Backups. To use a compliant backup created using the RDB approach, we would use the standard database backup restore process. This method showed a relative performance of 0.96. With the partitioning approach, one has to recombine the partitioned files and `INSERT` the records back into the original tables after restoring the database. Regardless, this still incurred a relative performance of 0.96. Both are faster than the cryptographic erasure approach by a statistically significant margin. This is primarily due to the cost of decrypting all encrypted values in the shadow tables before re-`INSERT`ing them back into the database's active tables. We believe that this could be an acceptable trade-off due to the relative infrequency in executing a complete restore of databases in an organization (even compared to other backup steps such as creation of backups).

5 Discussion

IT departments have multiple requirements to contend with in their operations, including compliance with purging and retention policies, keeping backups safe and in sync for a DR event, as well as maintain the application's performance.

Many organizations may choose, or are required, to leverage multiple backup sites that are geographically diverse to ensure redundancy as part of the DR and BC plan. Loss of data may mean the end of business, so these plans become essential for the business' survival. A business may also be subject to regulatory requirement to have redundant backups a certain number miles away from the main site; for example the SEC recommends "geographically dispersed back-up sites" [1]. Therefore, backups (either tape or alternative storage mediums) and data replication have to be moved offsite. In lieu of leasing or building a data center and the corresponding infrastructure, one might decide to utilize a cloud provider for their remote storage, or a 3rd party such as Iron Mountain [25]. Regardless of where the backup is located, retrieving and editing backups is a high-overhead process (see Section 4).

Although organizations may use asynchronous or synchronous replication, such methods are not a replacement for backups. Consider a disaster where logic corruption occurs, or a user deletes production data (whether accidentally or maliciously). The data would be affected on the primary copy and this corruption would then be faithfully replicated to other locations. Thus, storage based replication should not be used instead of a backup where numerous copies of the data are kept in case the current data is damaged or lost for any reason. In other words, backups are irreplaceable for a recovery effort.

With most systems, there are a multitude of ways to generate backups. If the organization's backup process generates a snapshot of a VM, none of the frameworks will be able to enforce purging compliance, due to all data being backed up (e.g., non-encrypted data, forensically recoverable data). This also applies if one were to snapshot an entire filesystem. While there are methods for segregating database tables into separate tablespaces, which can be on a separate filesystem, designing such mechanism is outside the scope of this paper.

The cryptographic erasure framework's use of encryption allows it to purge data from backups which are not physically accessible. Alternatively, using the RDB or partitioning approaches requires the backups to be accessible. Thus, organizations that disconnect backups for storage would be required to change their backup storage processes to accommodate the RDB or partitioning compliance process. Moreover, if an organization compliance requires it to manually connect offline backups to execute the deletion steps, the organization will have to a) regularly connect the storage and execute the compliance steps or b) wait until the purging period has passed before executing the backup compliance steps (resulting in a period of non-compliance).

Many organizations leverage incremental and delta backups, neither of which are supported by RDB or partition backup. Thus, any organization using either approach would be relegated to only using full backups. However, for any solution to be accepted by an industry organization, the ability to continue to use their existing backup processes is critical.

In sum, all of the retention and purging solutions require additional steps to be integrated into the existing procedures to enforce and maintain compliance. Where to apply these steps to strategically distribute the overhead impact will be at the discretion of the DBA when deciding which approach to implement.

In our evaluation, we are writing to a single SSD drive which has a limitation for the read and write speeds of roughly 500MB per second; other hardware of our server can be a limiting factor (e.g., bus speed, SATA controller). If one were to run the same experiment using enterprise hardware (storage and/or server) or an HDD, the times will change accordingly based on the hardware limitations. For example, if we were to use an SSD storage array connected to our server via multiple 32Gbps Fibre Channel ports, multipath, the I/O throughput and IOPS would dramatically increase, despite running on the same physical server. By reporting overheads as a

normalized rather than absolute runtime, we are able to mitigate the overall impact of hardware choices in our analysis.

6 Conclusion

IT Operations and the database administrator have competing priorities that they must consider. They must ensure that the application is performant, recovery is available, and that the data housed in the RDBMS is compliant with the laws under applicable jurisdiction. In this paper, we investigated the overheads associated with these considerations.

We examined several different methods for retaining data as well as purging the data across backups. Across different compliance techniques, we discussed how various factors contribute to the performance overhead when implementing a compliance system. With respect to retention, we showed that retention blocking approach offers a lower overhead compared to a retention archiving solution. However, the blocking approach may require a significant amount of code and process modification in the organization.

In purging data from backups, we compared three different methods : 1) restoring the backups, deleting the data, and recreating the backup, 2) partitioning the backups into segments that can be erased, and 3) cryptographically erasing data. The restore, delete, backup approach potentially involves a large amount of time between recalling the backup (if offsite), restoring the backup, deleting the purgeable data, and recreating the backup. This method is impractical in a large enterprise with hundreds or thousands of databases. Furthermore, an organization could not claim that the backups were not altered in case of a legal proceeding. For backup partitioning, DBA would have to suspend the application writing to the database in order to take a consistent backup of the database; thus, this solution is unrealistic in the era of always on and on-demand. Lastly, we have considered the cryptographic erasure approach in which the data can be purged from backups without recalling or reloading the backups by destroying the corresponding encryption keys. While the cryptographic restore process has significantly higher overhead compared to other methods, the backup restore process should be invoked infrequently.

It is up to each organization to determine which approach (with respect to performance, implementation, and associated limitations or costs) to choose in order to guarantee that privacy compliance is maintained. In this paper, we have evaluated the trade-offs in achieving these overarching goals.

Acknowledgments

This work was partially funded by US National Science Foundation Grant IIP-2016548, CME Group, and Argonne National Laboratory. Argonne National Laboratory's work was supported by the U.S. Department of Energy, Office of Science, under contract DE-AC02-06CH11357.

References

[1] 2003. Interagency Paper on Sound Practices to Strengthen the Resilience of the U.S. Financial System; Release No. 34-47638; April 7, 2003; Business Continuity Planning, BCP. https://www.sec.gov/news/studies/34-47638.htm
[2] 2019. https://about.att.com/story/2019/climate_resiliency_project.html
[3] 2020. https://merchantcostconsulting.com/lower-credit-card-processing-fees/how-to-increase-credit-card-authorization-rates/
[4] 2021. https://www.blancco.com/wp-content/uploads/2021/07/u-s-state-specific-data-disposal-laws.pdf
[5] 2022. Qcow. https://en.wikipedia.org/wiki/Qcow.
[6] 2023. https://www.ibm.com/products/instana/application-performance-monitoring
[7] 2023. https://www.ibm.com/products/db2-data-management-console
[8] 2024. https://www.cio.com/article/274740/outsourcing-sla-definitions-and-solutions.html
[9] Ahmed A Ataullah, Ashraf Aboulnaga, and Frank Wm Tompa. 2008. Records retention in relational database systems. In *Proceedings of the 17th ACM conference on Information and knowledge management*. 873–882.
[10] Centers for Medicare & Medicaid Services. 1996. The Health Insurance Portability and Accountability Act of 1996 (HIPAA).
[11] Steven R Christensen and Lawrence L Schkade. 1987. *Financial and functional impacts of computer outages on businesses*. Rothstein Associates.
[12] SD New York Dist. Court. 2005. Zubulake v. UBS WARBURG LLC. F. Supp. 2d, Volume 382, Page 536. , 536 pages. https://sosmt.gov/wp-content/uploads/attachments/E-ZubulakeV.pdf?dt=1519325634100
[13] Dennis F Galletta, Raymond Henry, Scott McCoy, and Peter Polak. 2004. Web site delays: How tolerant are users? *Journal of the Association for Information Systems* 5, 1 (2004), 1–28.
[14] John A Hoxmeier and Chris DiCesare. 2000. System response time and user satisfaction: An experimental study of browser-based applications. (2000).
[15] Windsor W Hsu, Alan Jay Smith, and Honesty C. Young. 2001. Characteristics of production database workloads and the TPC benchmarks. *IBM Systems Journal* 40, 3 (2001), 781–802.
[16] World Leaders in Research-Based User Experience. [n. d.]. Website response times. https://www.nngroup.com/articles/website-response-times/
[17] International Data Sanitization Consortium. 2017. Data Sanitization Terminology and Definitions. https://www.datasanitization.org/data-sanitization-terminology/
[18] 2012 4:25 pm UTC Jon Brodkin Oct 30. 2012. Hurricane Sandy takes data centers offline with flooding, power outages. https://arstechnica.com/information-technology/2012/10/hurricane-sandy-takes-data-centers-offline-with-flooding-power-outages/
[19] Krishna Kant. 2009. Data center evolution: A tutorial on state of the art, issues, and challenges. *Computer Networks* 53, 17 (2009), 2939–2965.
[20] KVM. [n. d.]. Linux KVM Documentation. https://www.linux-kvm.org/page/Main_Page.
[21] Ben Lenard, Alexander Rasin, Nick Scope, and James Wagner. 2021. What is lurking in your backups? *ICT Systems Security and Privacy Protection IFIP Advances in Information and Communication Technology*, 401–415.
[22] Jun Li, Sharad Singhal, Ram Swaminathan, and Alan H Karp. 2012. Managing data retention policies at scale. *IEEE Transactions on Network and Service Management* 9, 4 (2012), 393–406.
[23] Rich Miller. 2024. How sandy has altered data center disaster planning. https://www.datacenterknowledge.com/data-center-site-selection/how-sandy-has-altered-data-center-disaster-planning
[24] John Moore, Stephen J. Bigelow, and Paul Crocetti. 2022. What is BCDR? business continuity and Disaster Recovery Guide. https://www.techtarget.com/searchdisasterrecovery/definition/Business-Continuity-and-Disaster-Recovery-BCDR
[25] Iron Mountain. 2023. Offsite tape vaulting - secure storage. https://www.ironmountain.com/services/offsite-tape-vaulting
[26] Craig S. Mullins. 2021. Things the DBA hears. https://www.dbta.com/Columns/DBA-Corner/Things-the-DBA-Hears-147581.aspx
[27] Paulzipblog. 2023. Flashback Data Archive – Auditing Table data changes, a better approach. https://paulzipblog.wordpress.com/2021/05/03/auditing-table-data-changes-a-better-approach-flashback-data-archive/
[28] Mary K. Pratt and Melanie Luna. 2022. What is Data Stewardship? - definition from techtarget.com. https://www.techtarget.com/searchdatamanagement/definition/data-stewardship
[29] Nick Scope, Alexander Rasin, Ben Lenard, Karen Heart, and James Wagner. 2022. Harmonizing Privacy Regarding Data Retention and Purging. In *Proceedings of the 34th International Conference on Scientific and Statistical Database Management*. 1–12.
[30] Nick Scope, Alexander Rasin, Ben Lenard, and James Wagner. 2023. Compliance and Data Lifecycle Management in Databases and Backups. In *International Conference on Database and Expert Systems Applications*. Springer, 281–297.
[31] Nicholas Scope, Alexander Rasin, Ben Lenard, James Wagner, and Karen Heart. 2021. The Life of Data in Compliance Management. In *CYBER 2021: The Sixth International Conference on Cyber-Technologies and Cyber-Systems*. Springer.
[32] Nick Scope, Alexander Rasin, James Wagner, Ben Lenard, and Karen Heart. 2021. Database Framework for Supporting Retention Policies. In *International Conference on Database and Expert Systems Applications*. Springer. (to appear).
[33] Nick Scope, Alexander Rasin, James Wagner, Ben Lenard, and Karen Heart. 2021. Purging Data from Backups by Encryption. In *International Conference on Database and Expert Systems Applications*. Springer. (to appear).

[34] Robert Sheldon, Paul Kirvan, and Carol Sliwa. 2024. What is Business Impact Analysis (BIA)?: Definition from TechTarget. https://www.techtarget.com/searchstorage/definition/business-impact-analysis
[35] TPC. 2017. TPC-H Benchmark Database. https://www.tpc.org/tpch/.
[36] United States Congress. 1948. 28 U.S. Code §1732 Record made in regular course of business; photographic copies.
[37] Jurg van. Vliet, Flavia Paganelli, and Jasper Geurtsen. 2012. https://docs.aws.amazon.com/aws-backup/latest/devguide/deleting-backups.html
[38] James Wagner, Alexander Rasin, and Jonathan Grier. 2015. Database forensic analysis through internal structure carving. *Digital Investigation* 14 (2015), S106–S115.
[39] James Wagner, Alexander Rasin, and Jonathan Grier. 2016. Database image content explorer: Carving data that does not officially exist. *Digital Investigation* 18 (2016), S97–S107.
[40] James Wagner, Alexander Rasin, Karen Heart, Tanu Malik, and Jonathan Grier. 2020. DF-toolkit: interacting with low-level database storage. *Proceedings of the VLDB Endowment* 13, 12 (2020).
[41] James Wagner, Alexander Rasin, Tanu Malik, Karen Heart, Hugo Jehle, and Jonathan Grier. 2017. Database Forensic Analysis with DBCarver. (2017).
[42] Workspace. 2012. What is the true cost of lost data to business? https://www.workspace.co.uk/content-hub/business-insight/opinion-what-is-the-true-cost-of-lost-data-to-bus