Warehousing Data for Brand Health and Reputation with AI-Driven Scores in NewSQL Architectures: Opportunities and Challenges

Paulo Siqueira[1,2][a], Rodrigo Dias[1][b], João Silva-Leite[2][c], Paulo Mann[3][d], Rodrigo Salvador[2][e], Daniel de Oliveira[2][f] and Marcos Bedo[2][g]

[1] *Wikki Brasil, Parque Tecnológico, Rio de Janeiro/RJ, Brazil*

[2] *Institute of Computing, Fluminense Federal University, Niterói/RJ, Brazil*

[3] *Institute of Mathematics and Statistics, Rio de Janeiro State University, Rio de Janeiro/RJ, Brazil*

{paulo.costa, rodrigo.dias}@wikki.com.br, paulo.mann@ime.uerj.com.br,

Keywords: Multidimensional Model, Data Warehousing, Public Relations, Brand Monitoring, NewSQL, HTAP.

Abstract: This study explores the use of NewSQL systems for brand health and reputation analysis, focusing on multidimensional modeling and Data Warehouses. While row-based and relational OLAP systems (ROLAP) struggle to ingest large volumes of data and NoSQL alternatives rely on physically coupled models, NewSQL solutions enable Data Warehouses to maintain their multidimensional schemas, which can be seamlessly implemented across various physical models, including columnar and key-value structures. Additionally, NewSQL provides ACID guarantees for data updates, which is instrumental when data curation involves human supervision. To address these challenges, we propose a Star schema model to analyze brand health and reputation, focusing on the ingestion of large volumes of data from social media and news sources. The ingestion process also includes rapid data labeling through a large language model (GPT-4o), which is later refined by human experts through updates. To validate this approach, we implemented the Star schema in a system called `RepSystem` and tested it across four NewSQL systems: Google Spanner, CockroachDB, Snowflake, and Amazon Aurora. An extensive evaluation revealed that NewSQL systems significantly outperformed the baseline ROLAP (a multi-sharded PostgreSQL instance) in terms of: *(i)* data ingestion time, *(ii)* query performance, and *(iii)* maintenance and storage. Results also indicated that the primary bottleneck of `RepSystem` lies in the classification process, which may hinder data ingestion. These findings highlight how NewSQL can overcome the drawbacks of row-based systems while maintaining the logical model, and suggest the potential for integrating AI-driven strategies into data management to optimize both data curation and ingestion.

1 INTRODUCTION

The Multidimensional model serves as the logical foundation for OLAP (Online Analytical Processing) systems and is widely used in its materialized form within physically *row-based* database management engines that support Data Warehouses (ROLAP) (Morfonios et al., 2007; Zhang et al., 2019; Dehne et al., 2003). Although such engines are designed to handle efficiently normalized, day-to-day transactional tasks that follow the store-and-query pipeline, they struggle in the ingestion of big data as data typically undergoes a validation workflow to ensure ACID (Atomicity, Consistency, Isolation, and Durability) properties (Garcia-Molina et al., 2008). This limits the practical use of multidimensional models in favor of quick-ingestion NoSQL strategies with physically-coupled schemas that, in their turn, may be tricky and difficult to migrate between platforms (Ramzan et al., 2019; Chevalier et al., 2015).

The emergence of NewSQL solutions provides scalable executions for the mixed HTAP (Hybrid Transactional/Analytical Processing) paradigm (Valduriez et al., 2021; Huang et al., 2020; Grolinger et al., 2013). They enable Data Warehouses to continue being specified through multidimensional

[a] https://orcid.org/0000-0003-3652-284X
[b] https://orcid.org/0000-0003-1288-3093
[c] https://orcid.org/0009-0001-7881-6566
[d] https://orcid.org/0000-0002-3597-9170
[e] https://orcid.org/0009-0001-1202-5585
[f] https://orcid.org/0000-0001-9346-7651
[g] https://orcid.org/0000-0003-2198-4670

Siqueira, P., Dias, R., Silva-Leite, J., Mann, P., Salvador, R., de Oliveira, D. and Bedo, M.
Warehousing Data for Brand Health and Reputation with AI-Driven Scores in NewSQL Architectures: Opportunities and Challenges.
DOI: 10.5220/0013279800003929
Paper published under CC license (CC BY-NC-ND 4.0)
In *Proceedings of the 27th International Conference on Enterprise Information Systems (ICEIS 2025) - Volume 1*, pages 51-62
ISBN: 978-989-758-749-8; ISSN: 2184-4992
Proceedings Copyright © 2025 by SCITEPRESS – Science and Technology Publications, Lda.

designs and be implemented transparently across different physical models, including columnar and key-value schemas, and ensure ACID properties to transactional-like operations, which is ideal for OLAP systems requiring data curation (Chereja et al., 2021).

Quality in evolving data, quick ingestion, and fast responses to OLAP queries are particularly relevant requirements in brand health and reputation monitoring tools. In this context, companies can analyze and assess topic trends, perform sentiment analysis, evaluate the impact of stakeholders' statements, and measure engagement generated by digital influencers. Such systems require fast responsiveness from social media, print, and digital press to track positional trends over time (Doorley and Garcia, 2015). Accordingly, decision-making processes focus on acting and reacting to preserve or enhance the core values and perceived quality associated with each brand (Wang et al., 2021). The advantages of a strong reputation are substantial and wide-ranging: organizations with solid reputations attract top talent, secure better terms with suppliers, and increase their margins for PR-based products and operations (Wang et al., 2021).

Although reputation may seem like an intangible resource, it can be quantified and monitored through various metrics and analytical methods, such as measuring the reach and practical impact of a report (or social media post). This capacity to assess reputation enables companies to objectively evaluate, track, and strategically manage their position over time, where the present is exponentially more relevant than the past. This scenario fits perfectly into a multidimensional model, where dimensions represent data sources, authors, stakeholders, and metrics directly collected from the impact of posts and articles (hereafter referred simply to as *mentions*) that mention the company's brands. The time-sensitive nature of OLAP analysis enables grouping *mentions* into *hot* and *cold* portions based on their origin date within a pipeline that also aligns with the multidimensional structure. However, the expected volume of ingested data also suggests *row-based* solutions are unsuitable for the problem, whereas NoSQL solutions require yet another design that may be hard to migrate on-premise. The NewSQL approach is still unexplored for reputational analyses, to the best of the authors' knowledge (Li and Zhang, 2022; Wang et al., 2021).

In this study, we fill this gap by examining the behavior of different NewSQL systems considering a single Star schema designed for reputation and brand health analysis. This challenge involves not only a high volume of data ingestion from social media and press outlets but also requirements where *(i)* mentions must be quickly labeled regarding several facets in a single data pass, which we accomplish by using an LLM-based classification strategy and *(ii)* dimension values, new mentions, and data classification may eventually be refined by a human expert through update and removal operations that require consistency.

Accordingly, we implemented the multidimensional model as a system called `RepSystem` and instantiated it in four different NewSQL solutions with varying materialized physical implementations: Google Spanner, CockroachDB, Snowflake, and Amazon Aurora. Regarding data classification, we addressed the challenge of quantifying corporate reputation by designing a multi-purpose prompt that discretizes the value of each mention based on: *(i)* sentiment analysis (with five distinct labels), and *(ii)* top-k topics (drawn from a predefined list). Following an extensive experimental evaluation, we identified significant differences between the systems in terms of: *(i)* data ingestion time, *(ii)* storage requirements, *(iii)* OLAP query time, and *(iv)* maintenance costs. The results further indicated that: *(i)* the analyzed systems substantially outperform the baseline ROLAP implementation, which utilizes a PostgreSQL with multiple shards, and *(ii)* the primary bottleneck in the `RepSystem` is not ingestion nor query performance in NewSQL, but rather data classification.

The remainder of this paper is structured as follows: Section 2 presents the main concepts and related work. Section 3 introduces `RepSystem` and describes the materials and methods. Section 4 provides the experimental evaluation and discusses the key findings regarding the observed performance. Finally, Section 5 concludes the study and outlines directions for future work.

2 RELATED WORK

Business Intelligence Applied to Public Relations (PR) and Social Media. PR enterprises manage clients' reputations aiming at developing robust communicative strategies (Civelek et al., 2016). Data is sourced from social media platforms and the print and digital press, each impacting different audience profiles and shaping stakeholders' perceptions. Before the advent of advanced AI models, public relations depended on human content analysis and annotation to extract more information, including the identification of topics and the interpretation of the context surrounding each mention (Macnamara, 2005). Consequently, a set of key proprietary performance indicators (KPIs) was established for individual companies, typically integrating audience reach and impact into dashboard monitors that evolve over time (Marr and Schiuma, 2003). However, the manual nature of this process limited the scale of insights, as it excluded the analysis of vast volumes of digital mentions (Fan and Gordon, 2014). AI-powered systems can automatically label topics and assess sentiment, which enables PR professionals to focus on tracking insights related to brand reputation and detecting crises in their early stages. Despite these advancements, two significant issues are still open: *(i)* how to efficiently handle a large volume of mentions while maintaining system performance and KPI consistency over time, and *(ii)* how do those data management practices align with AI-based solutions to ensure that the ingestion process evolves consistently and scales with input?

Multidimensional Logical Model. The multidimensional model is a logical abstraction that represents data structures in a way that facilitates efficient querying and analysis. It organizes data into *dimensions* and *facts*, with dimensions representing descriptive attributes (*e.g.*, time, publishers) and facts representing quantitative metrics (*e.g.*, views, comments) (Inmon, 2005; Kimball and Ross, 2013; Golfarelli and Rizzi, 2009). Such tables are often structured as a Star schema, where a central fact table is connected to dimension tables, or a snowflake schema, where dimension tables are normalized. The model is typically implemented using a *materialized* strategy, where data is copied and pre-aggregated, and stored in a Data Warehouse within a DBMS.

Data Warehousing (DW). The DW approach is a structured process that materializes data from multiple sources to support business intelligence. It consists of four key layers that include *(i)* the mapping layer of external sources (providing raw input data), *(ii)* the stagging layer of data extracted from external sources, which serves as an intermediary area where data is standardized, cleaned, and aggregated by ETL (Extract, Transform, Load) routines, *(iii)* the warehouse layer, where materialized data are stored, and *(iv)* the presentation layer, which interfaces with end-users through BI tools and dashboards. Specifically, the third layer provides a physical implementation for the multidimensional model, which can take many forms (Inmon, 2005).

DBMS Physical Models. In addition to the traditional row-based implementation of the multidimensional model in relational DBMSs (ROLAP), other physical models, including NoSQL approaches, can be used for implementing the Data Warehouse layer (Cuzzocrea et al., 2013). This allows for the use of distributed query engines, enabling near-real-time processing with improved performance. However, the challenge lies in mapping the logical model to specific physical structures and query languages, which complicates data migration and tracking schema evolution. Distributed databases, such as NewSQL and HTAP systems, also have the potential to implement the multidimensional model across nodes or clusters using sharding. These systems not only overcome the limitations of previous approaches but also ensure a unified query language and deliver high availability and data consistency (Huang et al., 2020; Valduriez et al., 2021).

PostgreSQL. PostgreSQL is a row-based, relational DBMS that can implement a multidimensional logical model in a ROLAP architecture by storing dimensional and fact tables as clustered and indexed records. Records can be queried using high-level SQL, with extensions for multidimensional operators, such as `ROLLUP`. While query performance can be significantly improved with indexing and sharding, these techniques require expertise and monitoring to maintain efficiency. Despite these limitations, PostgreSQL provides a flexible, cost-effective baseline solution for systems that do not need the specialized functionality of a dedicated OLAP engine – particularly when the ingestion window can be relaxed.

Amazon Aurora. Aurora extends PostgreSQL's basic principles to the AWS environment, providing a distributed, fault-tolerant storage architecture. The storage mechanism uses six shards across different availability zones with auto-scaling and load balancing. Replication is log-based and takes full advantage of AWS resources, providing low-latency updates, synchronous writes, and asynchronous reads. Aurora fully supports SQL and the PostgreSQL wire protocol, optimizing query performance with adaptive caching and auto-scaling distributed in-memory search operators (Verbitski et al., 2017).

Google Spanner. Spanner is a distributed NewSQL system that scales horizontally. It relies on a unique architecture based on proprietary synchronized clocks (the Google TrueTime API) that ensures ACID properties with distributed transactions. Spanner replication and sharding are based on user-defined thresholds and occur automatically, enabling auto load balancing and reducing maintenance overhead. Spanner's shards communicate through the Paxos protocol replicated in multiple zones for redundancy. In terms of SQL support, Spanner offers full ANSI SQL compliance for data querying so that results can

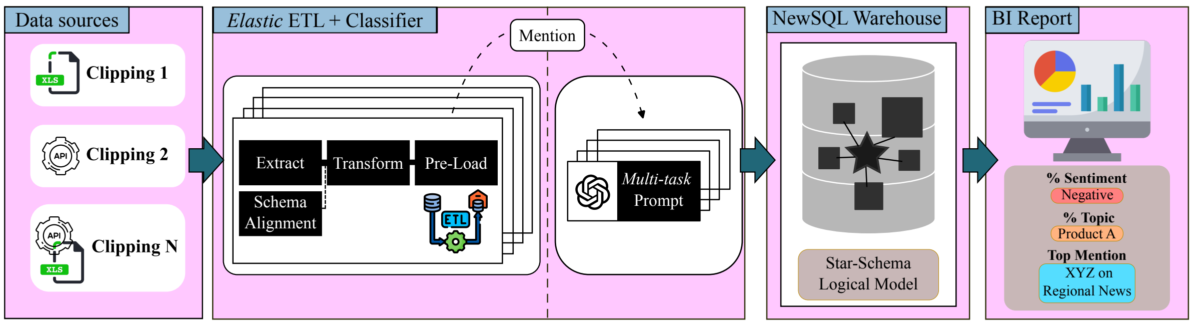

Figure 1: `RepSystem` pipeline for PR analysis (social media, and digital and printed press). (a) Sources, (b) Workers for ETL, (c) Data storage, and (d) BI report. The insertion features a multidimensional model implemented over a NewSQL DBMS.

be combined with multiple resources from Google Cloud services (Corbett et al., 2013).

CockroachDB. CockroachDB is a NewSQL built on a shared-nothing architecture that relies on the Raft consensus protocol to deliver ACID transactions. Isolation is achieved by a two-phase commit strategy, while a Multi-Version Concurrency Control mechanism manages the concurrent transaction schedule. Its physical model is a key-value storage with automatic sharding and geographical balancing, where entries can have multiple keys. The DBMS is fully compatible with PostgreSQL, including the Wire protocol and the structure of procedures. CockroachDB engine optimizes query execution through distributed planning, and the execution can be integrated with Kubernetes for orchestration, enabling seamless deployment and scaling in cloud-native environments (Taft et al., 2020).

Snowflake. Snowflake uses a multi-cluster architecture to replicate data across multiple regions to ensure availability. It uses transaction logs for durability and micro-partitioned storage to optimize performance and maintain consistency, which enables efficient data pruning and reduced scan times. Snowflake is SQL-compliant and a cloud-native DBMS for warehousing, separating workloads for query and storage to achieve independent scaling. Optimization is achieved through virtual abstractions applied to micro-partitions, enabling isolated compute resources that scale automatically. Those abstractions benefit from zero-copy cloning, which allows for fast and cost-effective data duplication without physically copying the data (Dageville et al., 2016).

Related Work. Data-driven approaches and AI, particularly Large Language Models (LLM), have transformed the PR ecosystem, especially in reputation management. This shift calls for new strategies and tools in the form of a hybrid intelligence that combines traditional PR with data-driven insights (Santa Soriano and Torres Valdés, 2021). The study of Jeong and Park (2023) shows PR efficiency enhancement with AI. An example of this use is modeling sentiment towards brands, as shown in the studies of Zhao et al. (2023) and Ingole et al. (2024), which provide accurate brand rankings and sentiment predictions. The work of Alqwadri et al. (2021) extends sentiment prediction to forecast consumer reliability, while Ur Rahman et al. (2022) applies a deep learning-based solution to enhance accuracy.

While these studies emphasize the data-driven potential for PR applications, they overlook how to manage the large volumes needed to realize these advancements. This study addresses these challenges by using NewSQL solutions to maintain the semantic integrity of KPIs and ensure evolving consistency through a detached multidimensional model. Additionally, we implement an LLM-based approach for scalable multi-task data labeling, advancing state-of-the-art reputation management.

3 MATERIAL AND METHODS

This section introduces `RepSystem`, a PR system that consolidates data from social media, digital, and printed press. We set `RepSystem` to use third-party services that collect data from original sources for generality sake and, as some data come from digitalized printed press, we use the umbrella term "clipping" to refer to that data gathering stage. Figure 1 presents the `RepSystem` data pipeline, following a materialized perspective.

Data Sourcing and Extraction. `RepSystem` was designed to handle inputs from files (spreadsheets), API outputs in .xml and .json, and a combination of all of the above. The extraction process for press content poses unique challenges due to the existence of paywalls, text varying sizes, and multimedia links. As for the third-party services, we connected `RepSystem` with one clipping tool for social media and four clipping applications that handle different media channels, including television, newspapers, and blogs. Although the filters that collect relevant mentions for every monitored brand were defined on the clipping setup, we also set a dynamic list of regular expressions (regex) terms associated with each brand, including company names, brands' names, and topics of interest to further filter spurious data from our experimental analysis.

Schema Alignment. The ETL process begins by mapping the dataset (rows and columns) to the logical model using a versioned YAML file that defines two parameters for each attribute of the model. The parameters are: *(i)* the original and mapped column names, and *(ii)* the extraction and transformation functions. Users can configure multiple functions per column, allowing each attribute to return specific types, such as a unique value, a list, or a multi-dimensional list. A try-catch mechanism ensures that the process iteratively applies each function per column in the order of priority defined by the written order. For instance, to extract the number of reactions, the YAML file can initially redirect to the function ext_reacts, which attempts to fetch the reaction value from a single column. If the function fails (*e.g.*, the column does not exist, or it is empty), the function ext_aggReacts is activated, which materializes the reaction values as a new column generated by the aggregate sum of comments, likes, and shares. A last priority, a default map function must also be defined to handle missing columns.

Data Transformation. Collected mentions were transformed to ensure standardization and prevent missing entries related to content, title, date, publisher vehicle, and author. The transformation pipeline consists of sequential steps (the *actions* mapped to functions), with a common yet critical component being the management of data inconsistencies using regular expressions (regex). We defined a hierarchical set of functions, ranging from exact two-way matches to approximate matches, so that, depending on the organization and structure of the input source, distinct regex strategies can be seamlessly applied.

Deterministic, Unique Identifiers. The ETL routines use a hashing method based on each mention's URL[1] combined with the channel name to create a unique identifier for each mention, *i.e.*,

[1]Digitalized contents were linked to cloud-stored files.

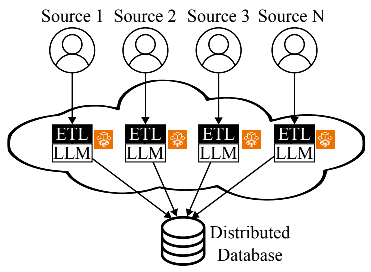

Figure 2: `RepSystem` data ingestion.

the idMention. This process relies on the SHA-1 algorithm combined with UUID5 casting to produce a deterministic 128-bit identifier, which, by definition, prevents duplicate inserts. Finally, the ETL combines this identifier with the company identifier, as different mentions may be related to distinct companies, *e.g.*, two competitor brands in the same article. This composite key is not only semantically meaningful but also eliminates the need for locks during insertion, enabling concurrent inserts and simplifying transaction scheduling.

Data Ingestion. Mentions are always loaded in append mode, whereas dimensions are loaded using a destructive merge policy. The ingestion flow follows the simplified diagram in Figure 2, assuming data are ingested upon user requests (without a predefined time window) Each request triggers a new containerized process that fetches a chunk of raw data from a clipper source, aligns and transforms the internal information, and sends the content for LLM-based analysis. The orchestration of this elastic approach can be managed using an external balancer or a simple mechanism, such as a message queue associated with a triggering lambda function. `RepSystem` uses transactional control tables to track the status of launched containers so that if the ETL process fails at any step, its status is recorded for reproducibility and debugging purposes.

LLM-based Classification. `RepSystem` analyses require data content to be evaluated regarding sentiment analysis (positive, neutral, negative) and top-k topic identification. Since these two analyses are different in nature, `RepSystem` relies on a multi-task prompting strategy to speed up the analysis, avoiding the problem of double-passing the data and reducing maintenance overhead. In a nutshell, `RepSystem` ETL simultaneously handles the two tasks, each associated with a set of mentions and a set of label pairs (sentiment, top-k topics). For each mention in a task, there is one corresponding label. Therefore, a task-shared prompt can be used to unify the represen-

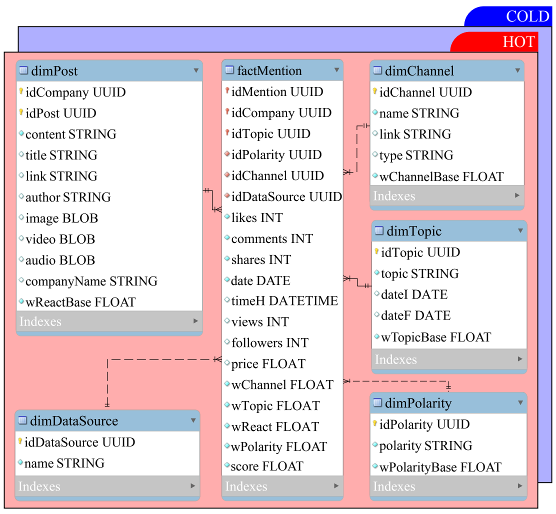

Figure 3: The simplified Star schema for `RepSystem`.

tation across all tasks. This single prompt template is applied to each mention, producing the outputs for the two tasks as pairs of labels. `RepSystem` uses a *zero-shot* approach alongside the multi-task strategy, which labels each input according to a pre-trained encoder for general-purpose PR analysis.

Discrete Prompting. To investigate a suitable template for the multi-task problem, we use discrete prompting as part of our prompt engineering strategy (Liu et al., 2023), which involves manually creating prompts using natural language. Accordingly, we first developed an empirical prompt using the chain-of-thought process and then generated three additional prompts. Next, we assessed the quality of the produced prompts by measuring the F1-Score on a curated and manually annotated dataset, and calculated the average and standard deviation to determine the most suitable prompt for data labeling.

Multidimensional Data Model. Figure 3 presents the `RepSystem` multidimensional model, which is a Star schema for the fact table factMention. The model is mirrored between the "hot" and "cold" portions, with data stored in the former part being the most recent and data in the latter part being archived. This separation occurs at the DDL schema level, enabling tuning and optimization of different workloads and indexing. The logical model contains five-dimensional tables, each serving a specialized role: *(i)* dimPost, which records mention content information (entries are relatively large); *(ii)* dimDataSource, which tracks the origin of each mention's data source, *e.g.*, social media, newspapers, TV, etc.; *(iii)* dimChannel, which enables the identification of the specific vehicle for the communication, *e.g.*, Social Net foo, TV foo, etc.; *(iv)* dimTopic, which structures the topics of interest related to each company; and *(v)* dimPolarity, which defines the values for sentiment analysis. Both dimTopic and dimPolarity dimensional values are used as lists of options by the LLM to solve the multi-task classification problem.

The fact table contains references to the dimensional tables, with the dimension date being implicitly captured by the fact table. The numerical metrics include the number of reactions (sum of likes, comments, and shares), views, followers (at the time of the mention), price (if it is a paid mention), and four scores: weights for polarity ([-1, 1]), channel, topic, and reactions, along with an overall aggregate *score* for the mention ([0, 1]). This score value was defined as a linear combination to emulate proprietary metrics, which have their own biases toward specific dimensions. In our evaluations, we calculate the score value as $score = w_p \cdot \sum_{i \in \{\text{Channel, Topic, React}\}} w_i$ with w_p being the polarity weight. Notice LLM-based classification may occasionally miss. Therefore, PR experts will review the process, correcting relevant misclassifications and triggering the adjustment of associated metrics. These adjustments will be consistently propagated across NewSQL shards without requiring modifications to the logical model.

`RepSystem` implementation. `RepSystem` was implemented in Python 3.10.13, venv environment, with packages numpy 1.26.4, pandas 2.1.4, selenium 4.19.0, Unidecode 1.38.4, beautifulsoup4 4.12.3, SQLAlchemy 2.0.30, boto3 1.34.161, PyYAML 6.0.1, tenacity 8.5.0. GPT 4o was accessed with package openai 1.30.3 using a corporate credential from the Microsoft AzureAI Cloud portal with the highest available service level agreement for a basic regular corporate account. `RepSystem` deployed on a `RepSystem` on AWS Cloud Platform using AWS Lambda with ECS running the ETL as isolated AWS fargates. The multidimensional logical model was instanced on several environments, with cloud-oriented instances (all East-US), namely *(i)* AWS RDS for Postgres, *(ii)* AWS Aurora, *(iii)* Google Spanner on Google Cloud Platform, *(iv)* CockroachDB serverless instance on proprietary cloud, and *(v)* Snowflake instance running on Amazon AWS.

Queries. The multidimensional model was described using SQL DDL and instantiated in each DBMS with minor adjustments for a fairer comparison (type compliance). All of the evaluated queries were expressed with SQL operators derived from the basic `CUBE` and `ROLLUP` multidimensional operators, which were unsupported by some DBMSs. Next, we define five

Table 1: Average F-Measure performance for zero-shoting GPT4o-mini regarding Single Tasks (two data passes) *vs.* Multi-Task (just one data pass) by using the best prompts constructed with discrete prompting.

Task	Strategy	Company A	Company B	Company C
Sentiment	Single-Task	.79	.81	.82
	Multi-Task	.82	.84	.87
Topic	Single-Task	.84	.64	.70
	Multi-Task	.82	.63	.67

queries that handle different workloads and produce significant insights into various branding reputation analyses by a senior PR expert, namely: **(Q1):** the proportion of mentions by polarity per company, rolling up on dimension date (lightweight workload – Overall sentiment towards the brand); **(Q2):** the average reaction by channel per company and polarity, rolling up on dimension date (medium workload – Inferred support or detractor channel); **(Q3):** the most frequent pairs ⟨company, topic⟩ over time, with a `CUBE` on topics (medium workload – Dominant narratives per period); **(Q4):** highest-ranked channels (sum of absolute scores) by company aggregate date per source, rolling up on dimension date (heavy workload – Channels impacting your brand); **(Q5):** top-k monthly and annual mentions by company, data source, polarity, top-k' topics, and top-k'' channels (heavy workload – Most influential content over the period by producer).

Whenever possible, we used SQL-only functions to reduce the writing, such as in **(Q2)**, which was reduced to a specific instance of the Market-Basket problem solved by the partition-based *A-Priori* solution. The same "function-packing" rationale was employed to `UNION` result for DBMSs not supporting the multidimensional operators. Finally, we wrote **(Q5)** with `LATERAL JOIN/UNNEST` to avoid converting window functions for specific SQL dialects.

4 EXPERIMENTAL EVALUATION

Dataset Workloads. We curated a dataset by selecting two years of data from three real-world companies across very different domains, totaling over two million human-annotated fact tuples (4.7GB of raw text). The dataset covers 12,975 channels, 18 distinct data sources, and 608 unique topics. We divided the dataset into 80 stratified workloads, with stratification based on the original "topic" as the meta attribute for each company. Next, we removed the sentiment and topic annotations and submitted the resulting batches for ETL processing, producing spreadsheets containing raw content from the clipping sources. Unlike existing HTAP performance benchmarks (Cole et al., 2011), this subject-oriented dataset incorporates an AI-driven annotation aspect into the OLAP problem.

Multi-Task Validation. To assess the viability of using the proposed multi-task prompting approach as part of `RepSystem`'s one-pass labeling stage in the ETL process, we compared this strategy with the baseline approach, which utilizes two specialized prompts (one for sentiment analysis and the other for top-1 topic classification). Table 1 presents the average F-Measure (a weighted score combining precision and recall) per company in the dataset workload, based on a zero-shot comparison between our multi-task approach and the baseline. This comparison was conducted using the GPT-4o-mini LLM (version gpt-4o-mini-2024-07-18) through the Azure OpenAI API, with a temperature setting of 0.3. The results show minimal differences in average performance between the single-task and multi-task strategies, suggesting that the proposed approach can effectively utilize a one-pass LLM-based analysis rather than issuing multiple requests to label each mention. Accordingly, we adopted the same zero-shot setup and prompt for the final part of our ETL evaluation in subsequent assessments.

Experimental Setup. The PostgreSQL setup was configured with 02 vCores, 4 GiB of RAM, and 32 GiB of storage, instantiated in the East-US AWS Cloud. For Google Spanner, a comparable configuration involved provisioning 500 PUs (0.5 nodes), providing approximately 02 vCPUs, 8 GiB of RAM, and 32 GB of storage. The AWS Aurora instance was configured with 02 vCPUs and 4 GiB of RAM (db.t3.medium class), with automatic storage scaling and failover across six availability zones. In Snowflake, we used a comparable configuration with a Small Virtual Warehouse, offering 02 vCores and 4 GiB of RAM, with automatic scaling and sharding. Finally, CockroachDB was configured with 02 compute units (CUs), suitable for moderate workloads in a serverless, self-managed environment. For Aurora, Snowflake, and CockroachDB, storage was set to scale automatically.

Classification Elapsed Time. We measured classification time separately to assess the individual impact

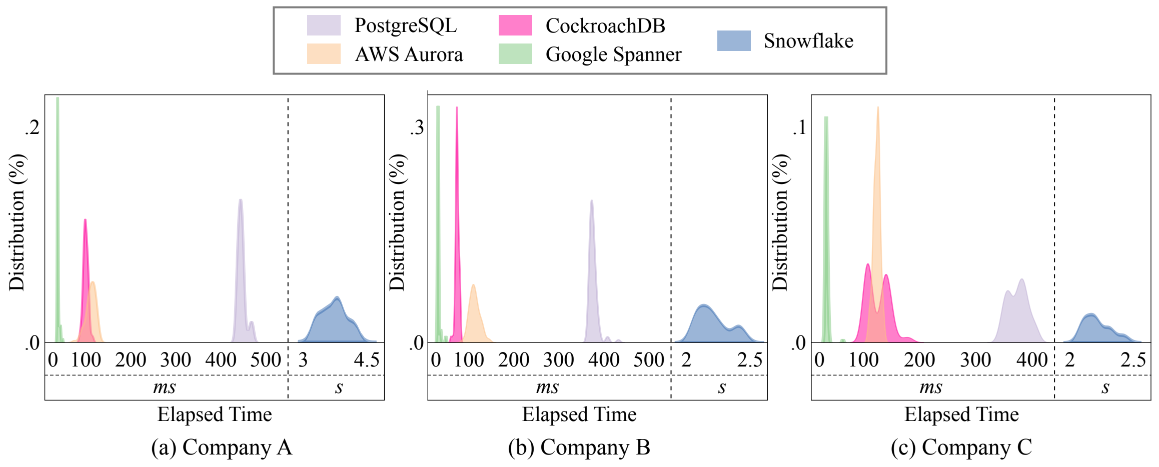

Figure 4: Distribution of elapsed time required for ingesting data batches into different NewSQL DBMSs. The time spent on LLM-based classification was subtracted from the total elapsed time.

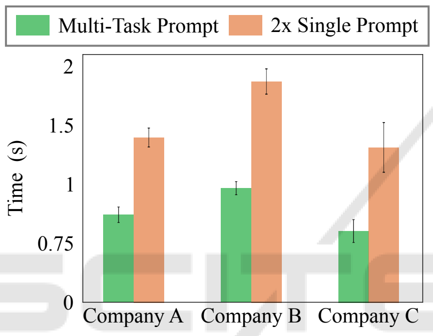

Figure 5: Elapsed time for multi-task analysis.

of this process on the ETL pipeline. Although we employed the highest available service level for the LLM (ensuring a throughput of 250k tokens per minute), our measurements indicated that processing times did not cause sufficient queuing to exceed this limit. Figure 5 presents the average elapsed time required to label a mention. Results show that the classification time using the multi-task prompt remained relatively stable across all companies examined, regardless of the specific topic list from which the prompt was expected to extract a response. These findings suggest that *(i)* data labeling via the LLM is both a flexible and scalable option for transformations involving multiple natural language analyses, and *(ii)* while scalable per request, classification time still significantly impacts the overall data ingestion time. Thus, we factored this time into the total duration of the remaining evaluations.

Data Ingestion. We ingested all data batches into each DBMS using parallel ETL workers instantiated as Fargate components in an AWS Elastic Container Service (ECS) cluster, with the maximum number of connections limited to 80 parallel transactions. Each mention was ingested individually (single inserts) to highlight the impact of transaction management for each compared approach. This stress injection mechanism resembles more closely an HTAP environment, where on-demand inserts can arrive individually. We carefully subtracted data labeling time from the analysis, as it is an expensive component of the ETL pipeline that could skew the timing results.

Figure 4 shows the elapsed time required to process the inputs and ingest data into each DBMS. PostgreSQL struggled with ingestion, as data mentions had to pass through relational constraint validation. Aurora exhibited improved performance by taking full advantage of inner sharding per company. It reduced the PostgreSQL insertion time by up to 83% (Company A) and maintained lower buffer usage, avoiding potential overflows. Although Snowflake showed the widest performance variation during ingestion, its performance was poor, as the tool is designed for bulk-loading data rather than individual inserts commonly found in OLTP environments. While outside the AWS cloud, CockroachDB outperformed Snowflake and AWS Aurora, on average. In particular, it was also 5× faster than baseline PostgreSQL. Finally, Google Spanner outperformed the competitors in data ingestion, requiring nearly 60% less time than the second-best competitor, on average.

Query Performance. We executed five separate queries for each data batch to evaluate query performance across the tested DBMSs. The results revealed significant variation in query response times and highlighted differences in elapsed times among the evaluated systems. Snowflake demonstrated the fastest query performance overall, outperforming PostgreSQL by up to ten times. This performance

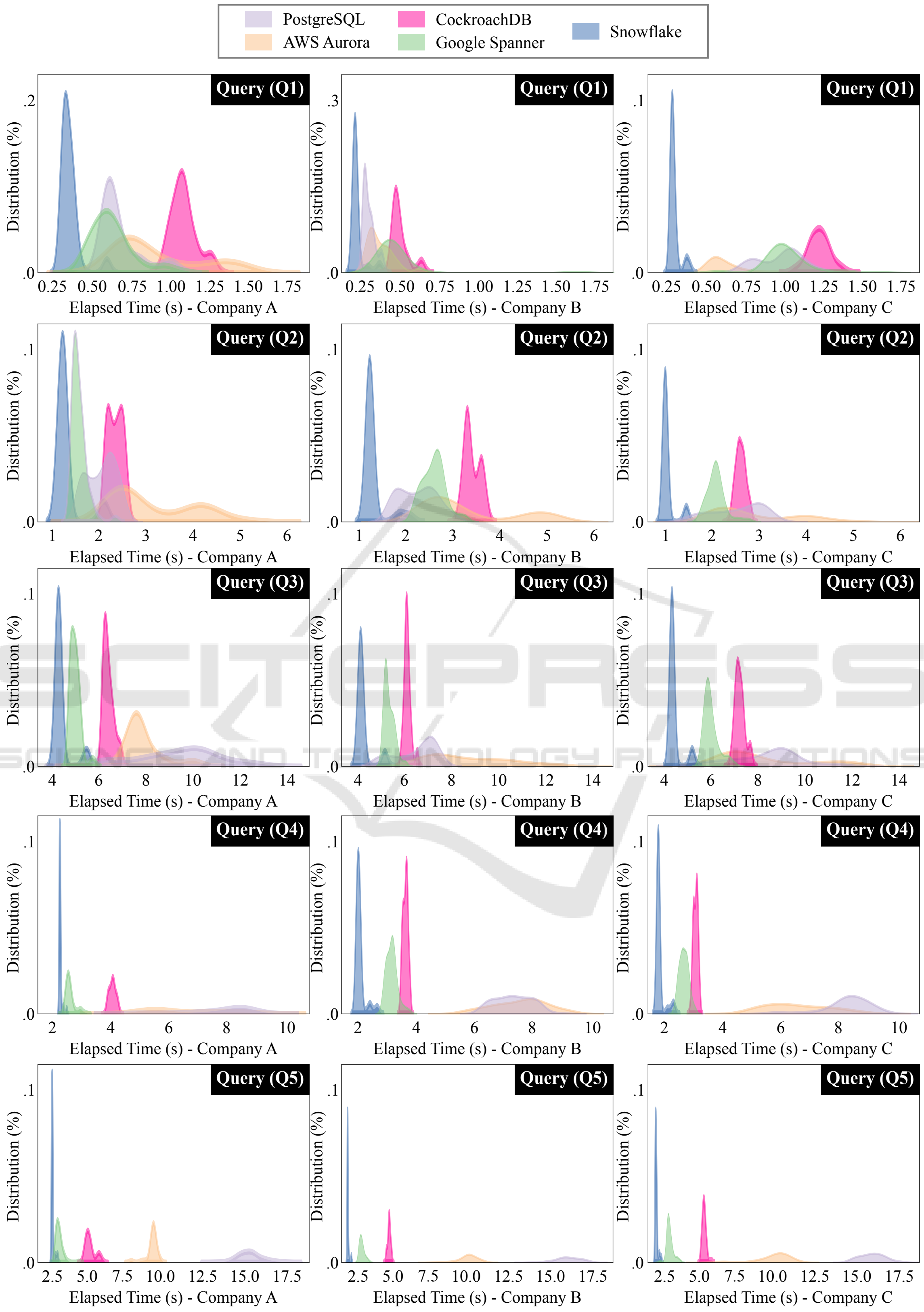

Figure 6: Distribution of elapsed time required for querying different companies. Line entries corresponds to queries and column entries are companies.

highlights Snowflake's highly optimized query execution engine that was particularly evident for queries **(Q4)** and **(Q5)**, which involve multiple nested/lateral joins. Google Spanner was the second-best performer, trailing Snowflake by up to 35%, but consistently surpassing the other systems regarding elapsed time for every examined query.

CockroachDB and AWS Aurora exhibited similar query performance, with CockroachDB substantially outperforming Aurora in complex queries. Despite their similarities, PostgreSQL's non-distributed architecture occasionally showed latency spikes, while AWS Aurora benefited from its high scalability and inner-sharding mechanisms for batch queries. The performance per query varied widely, with queries **(Q3)** to **(Q5)** posing more challenges across all DBMSs. These queries required up to 10× more time and memory, as observed in the panel insights for CockroachDB, Snowflake, and Spanner, suggesting that query complexity and data relationships heavily influenced execution times, regardless of the underlying cloud system. PostgreSQL struggled significantly with heavy workloads **(Q4)** – **(Q5)** due to the lack of distributed processing and limited parallelism in query execution. However, it managed lightweight workloads like **(Q1)** – **(Q2)** relatively well, highlighting its suitability for midsize query workloads.

Although the same multidimensional model was implemented by all DBMSs, *trade-offs* were observed between querying and ingestion. For instance, while Snowflake excelled in querying, its ingestion performance was outpaced by CockroachDB and Spanner, underscoring the importance of understanding the specific workload requirements when implementing the multidimensional model in a physical system.

Storage Requirements. Figure 7 shows the storage consumption across all DBMSs after processing and ingesting the mentions. As expected, PostgreSQL and AWS Aurora showed similar storage consumption, with Aurora using nearly half the space of PostgreSQL. While Snowflake relies on columnar storage and heavy data compression, it ranked last in overall storage cost due to metadata retention for each column and insert. CockroachDB outperformed PostgreSQL and Aurora in terms of storage management, requiring 5× and 1.3× less storage than PostgreSQL and Aurora, respectively. CockroachDB's controlled replication model provided additional resilience without significantly increasing storage usage, requiring less space than all the other systems. Finally, Google Spanner displayed storage optimization, requiring slightly more storage than CockroachDB on average. These

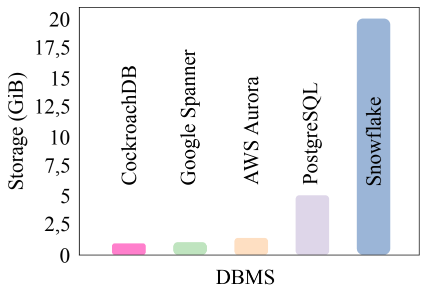

Figure 7: Overall storage demanded for each solution.

Table 2: *MedianRank* aggregation for the average performance of compared NewSQL approaches. The first position tie was broken by the next lowest ranking.

DBMS	**Ingestion**	**Search**	**Storage**	***MedianRank***
A. Aurora	3	4	3	3
G. Spanner	1	2	2	1
CockroachDB	2	3	1	2
Snowflake	4	1	4	4

findings indicate that Aurora mitigated PostgreSQL's traditional *row-based* storage costs, which were much different than the columnar model employed by Snowflake. Overall, Spanner and CockroachDB's distributed design offered robust performance, setting the benchmark for storage efficiency.

Comparison of NewSQL Approaches. We consolidated the average performance for ingestion, querying, and storage requirements into *rankings* (1 = best, 5 = worst) to aggregate the individual performances using the *MedianRank* strategy (Fagin et al., 2003). The query performance was ranked based on the average performance for queries **(Q3)** through **(Q5)**. Table 2 summarizes the ranking distribution and final positions, where Google Spanner and CockroachDB achieved the best performance across all three criteria for the evaluated PR data. Snowflake exhibited the largest *trade-off*, being the fastest query solution but with high storage costs. AWS Aurora ranked third overall, primarily struggling with data ingestion. In practice, however, budget and long-term maintenance are important constraints to address when planning project implementation.

Infrastructure Costs. Figure 8 presents a simplified cloud-based cost summary for the previous experiments. Since the comparison is cloud-dependent, we defined a *Relative Cost metric* that averages the total cost billed for the entire evaluation, divided by

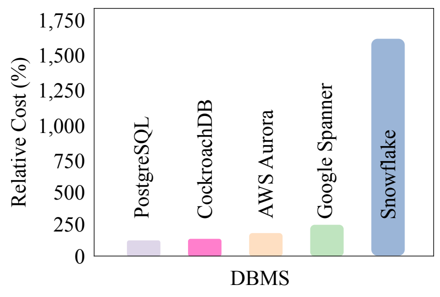

Figure 8: Comparison of scaled cloud-based costs.

the (fractional) number of hours it took to complete. We set PostgreSQL as the reference value (with 100% of Relative Cost) and then linearly scaled the costs of the remaining DBMSs in comparison to this reference. Self-hosted CockroachDB offered the lowest cost baseline, being the cheapest alternative among the compared NewSQL solutions. AWS Aurora optimized costs by adjusting resources based on the workloads but required a higher budget compared to PostgreSQL. Snowflake, with its flexible consumption-based pricing model, was the most expensive among the competitors due to struggles with individual data ingestion. Finally, Spanner's costs were proportionally higher than those of CockroachDB and Aurora, reflecting potential pricing differences between the host clouds.

Maintenance Issues. While PostgreSQL maintenance has native limitations that require human expertise for query and load tuning, AWS Aurora provides insights and maintenance warnings through the Amazon RDS Console, indicating workload bottlenecks and offering recommendations for query optimization. This proprietary cloud strategy is also observed in Google Spanner, which relies on Google Cloud monitoring and logging. Snowflake and CockroachDB offer their cloud-agnostic consoles that provide insights and can even automatically apply enhancements for workloads. Snowflake offers a more detailed analysis, as storage and queries are evaluated separately.

Discussion. The evaluation of a real-world dataset for brand monitoring highlighted performance and scalability differences across various NewSQL approaches implementing the same logical model for `RepSystem`. The findings showed that these approaches, particularly Google Spanner and CockroachDB, significantly outperformed traditional ROLAP-based systems in terms of data ingestion, even under the stress of single-insert operations. Optimized LLM-based classification in a single data pass was a bottleneck compared to data ingestion into the NewSQL DBMSs. Results indicated that Snowflake dominated query performance, with Spanner and CockroachDB also reaching competitive values. Regarding storage efficiency, CockroachDB performed the best, followed by Google Spanner.

If all performance criteria (ingestion, query performance, and storage) are equally weighted, Google Spanner emerges as the top contender, closely followed by CockroachDB. Moreover, when considering cost-effectiveness, CockroachDB, AWS Aurora, and Google Spanner offered the best value, demonstrating efficiency in handling large-scale deployments at competitive costs. Finally, for organizations with diverse cloud environments or multi-cloud strategies, CockroachDB and Snowflake provided superior flexibility, while Spanner and Aurora offered deeper integration within their respective cloud ecosystems. Accordingly, specific needs related to performance, cost, and flexibility will likely result in different NewSQL flavors implementing the same logical model.

5 CONCLUSION AND FUTURE WORK

We explored the use of NewSQL systems for brand health analysis, focusing on a logical multidimensional model across various physical implementations. We found that NewSQL systems, tackling the logical-physical model separation, can overcome the limitations of traditional ROLAP systems, simplify DBMS migration, and enhance schema evolution. We proposed a simplified Star schema for PR analysis, integrating data from social media, digital, and printed press through the `RepSystem` application. Unlike existing HTAP benchmarks, this approach required AI for multiple data labeling, which we addressed with a multi-task prompt strategy using a distributed LLM.

Our evaluation revealed performance differences across different NewSQL solutions, such as *(i)* Snowflake excelling in complex queries but struggling with data insertion and *(ii)* CockroachDB outperforming the competitors in storage efficiency but being less efficient than Google Spanner in heavy workloads. Overall, NewSQL systems outperformed the baseline ROLAP PostgreSQL, with data classification being the main identifiable bottleneck. Future work will include evaluating larger databases, simulating near-real-time PR data flows, and developing a benchmark for brand health and monitoring.

ACKNOWLEDGEMENTS

This study was financed by the Coordenação de Aperfeiçoamento de Pessoal de Nível Superior - Brasil (CAPES), Brazil - Finance Code 001, and by the Carlos Chagas Filho Research Support Foundation of the Rio de Janeiro State (FAPERJ), Brazil (Grants E-26/204.238/2024 and E-26/204.544/2024).

REFERENCES

Alqwadri, A., Azzeh, M., and Almasalha, F. (2021). Application of machine learning for online reputation systems. *IJAC*, 18(3):492–502.

Chereja, I., Hahn, S. M. L., Matei, O., and Avram, A. (2021). Operationalizing analytics with newsql. In *Soft. Eng. and Alg.*, pages 249–263. Springer.

Chevalier, M., El Malki, M., Kopliku, A., Teste, O., and Tournier, R. (2015). How can we implement a multidimensional data warehouse using nosql? In *EIS*, pages 108–130. Springer.

Civelek, M. E., Çemberci, M., and Eralp, N. E. (2016). The role of social media in crisis communication and crisis management. *IJRBSC*, 5(3).

Cole, R. et al. (2011). The mixed workload ch-benchmark. In *IWTDS*, pages 1–6.

Corbett, J. C. et al. (2013). Spanner: Google's globally distributed database. *ACM Trans. Comput. Syst.*, 31(3).

Cuzzocrea, A., Bellatreche, L., and Song, I.-Y. (2013). Data warehousing and olap over big data: current challenges and future research directions. In *DOLAP*, page 67–70. ACM.

Dageville, B. et al. (2016). The snowflake elastic data warehouse. In *ICMD*, page 215–226, NY, USA. ACM.

Dehne, F., Eavis, T., and Rau-Chaplin, A. (2003). Parallel multi-dimensional rolap index. In *ISCCG*, pages 86–93.

Doorley, J. and Garcia, H. F. (2015). *Reputation management: The key to successful public relations and corporate communication*. Routledge.

Fagin, R., Kumar, R., and Sivakumar, D. (2003). Efficient similarity search and classification via rank aggregation. In *SIGMOD*, pages 301–312.

Fan, W. and Gordon, M. D. (2014). The power of social media analytics. *Commun. ACM*, 57(6):74–81.

Garcia-Molina, H., Ullman, J., and Widom, J. (2008). *Database Systems*. PH, USA, 2 edition.

Golfarelli, M. and Rizzi, S. (2009). *Data Warehouse Design: Modern Principles and Methodologies*. McGraw-Hill Education.

Grolinger, K., Higashino, W. A., Tiwari, A., and Capretz, M. A. (2013). Data management in cloud environments: Nosql and newsql data stores. *J. of Cloud Computing: Advances, Sys. and App.*, 2:1–24.

Huang, D., Liu, Q., Cui, Q., Fang, Z., Ma, X., Xu, F., Shen, L., Tang, L., Zhou, Y., Huang, M., Wei, W., Liu, C., Zhang, J., Li, J., Wu, X., Song, L., Sun, R., Yu, S., Zhao, L., Cameron, N., Pei, L., and Tang, X. (2020). Tidb: a raft-based htap database. *PVLDB*, 13(12):3072–3084.

Ingole, A., Khude, P., Kittad, S., Parmar, V., and Ghotkar, A. (2024). Competitive sentiment analysis for brand reputation monitoring. In *ICETITE*, pages 1–7.

Inmon, W. (2005). *Building the Data Warehouse*. Wiley.

Jeong, J. Y. and Park, N. (2023). Examining the influence of artificial intelligence on public relations: Insights from the organization-situation-public-communication (ospc) model. *JCRI*, 9(7):485–495.

Kimball, R. and Ross, M. (2013). *The Data Warehouse Toolkit: The Definitive Guide*. Wiley.

Li, G. and Zhang, C. (2022). Htap databases: What is new and what is next. In *SIGMOD*, page 2483–2488, New York, NY, USA. ACM.

Liu, P., Yuan, W., Fu, J., Jiang, Z., Hayashi, H., and Neubig, G. (2023). Pre-train, prompt, and predict: A systematic survey of prompting methods in natural language processing. *ACM Comput. Surv.*, 55(9).

Macnamara, J. (2005). Media content analysis: Its uses, benefits and best practice methodology. *Asia-Pacific Public Relations J.*, 6:1.

Marr, B. and Schiuma, G. (2003). Business performance measurement – past, present and future. *Management Decision*, 41(8):680–687.

Morfonios, K., Konakas, S., Ioannidis, Y., and Kotsis, N. (2007). Rolap implementations of the data cube. *ACM Comput. Surv.*, 39(4):12–es.

Ramzan, S., Bajwa, I. S., Kazmi, R., and Amna (2019). Challenges in nosql-based distributed data storage: A systematic literature review. *Electronics*, 8(5).

Santa Soriano, A. and Torres Valdés, R. M. (2021). Engaging universe 4.0: The case for forming a public relations-strategic intelligence hybrid. *Public Relations Rev.*, 47(2):102035.

Taft, R. et al. (2020). Cockroachdb: The resilient geo-distributed sql database. In *SIGMOD*, page 1493–1509, NY, USA. ACM.

Ur Rahman, M. W., Shao, S., Satam, P., Hariri, S., Padilla, C., Taylor, Z., and Nevarez, C. (2022). A BERT-based Deep Learning Approach for Reputation Analysis in Social Media . In *AICCSA*, pages 1–8. IEEE.

Valduriez, P., Jimenez-Peris, R., and Özsu, M. T. (2021). *Distributed Database Systems: The Case for NewSQL*, pages 1–15. Springer.

Verbitski, A. et al. (2017). Amazon Aurora: Design considerations for high throughput cloud-native relational databases. In *SIGMOD*.

Wang, Y., Cheng, Y., and Sun, J. (2021). When public relations meets social media: A systematic review of social media related public relations research from 2006 to 2020. *Public Relations Rev.*, 47(4):102081.

Zhang, Y., Zhang, Y., Wang, S., and Lu, J. (2019). Fusion olap: Fusing the pros of molap and rolap together for in-memory olap. *TKDE*, 31(9):1722–1735.

Zhao, R., Gui, L., Yan, H., and He, Y. (2023). Tracking brand-associated polarity-bearing topics in user reviews. *TACL*, 11:404–418.