Chapter 1

Introduction

In the current technological world, data is envisaged as an authentic information used for various qualitative and quantitative analysis. In the past few decades, data is growing exponentially at an unprecedented scale in diverse formats such as structured, semi-structured, and unstructured in almost all domains. This exponentially increasing data is consistently processed via different stages of its life-cycle operations such as data generation, pre-processing, extraction, storage, transmission etc. This constantly emerging limitless data led to requirements of large storage capacity and computing power. Moreover, rapid evolution of social media and web-based communications have also led to phenomenal growth in human-generated semi-structured/unstructured data related to various fields of our day to day life. This exponential growth of data in various application domains such as social media, IoT, scientific applications etc., require an efficient data management. Although, the amount of data processed in these application domains are collected from various sources, but the major part of this data is generated from different transformations applied on the existing data. To measure the data quality [9], and its trustworthiness [47] for different perspectives such as audit trail, error tracing, data diagnostics, fact investigations [171], and rumour detection [147] etc., a substantial metadata [84], i.e., "descriptive information about data" is much required. In addition to these metadata, other significant information such as the owner of data, origin or direct/indirect sources of data, derivation process and history of data is also required. *Provenance data* is a kind of metadata information that gives the various significant information about any data object such as owner of data, its origin and direct/indirect sources, different

transformations applied on data, history of data updates etc [9, 2, 13, 149, 96]. In its simplest form, provenance can be described as a type of metadata about the data product. Alternatively, provenance can be described by answering the questions *how* and *where* the data is produced and *when* and by *whom*. Therefore, provenance information is much essential to comprehend the credibility and quality of data [30].

1.1 Importance of Data and Big Data

The 21st Century will be known as the century of Data as it has witnessed an unprecedented growth of data in almost all domains. According to an IDC report [151], by 2025 we will have approximately 163 Zettabytes of data in the world which is almost four times the current data. This phenomenal growth in data can be attributed to availability of cheap digital storage and to the advancement in data sensing technology.

The term "Big Data" was coined by John Mashey in 1990 [8], although it caught the fancy of the computing world in the early years of the present century. Big Data is characterised by 7 V's viz., Volume, Velocity, Veracity, Variety, Variability, Visualization, and Value. Big Data renders computing infrastructure inadequate in quick time. This can be attributed to increasing volumes of data coupled with shrinking response times. The challenges and opportunities with Big Data are very comprehensively compiled in a white paper by the leading computer scientists and industry researchers of the US in 2012 [101], a study initiated by Computing Research Association.

1.2 Journey of Database

1.2.1 SQL Database

Relational databases have been the mainstay of the data community for decades starting from mid 1980's. It was proposed by Edgar F Codd in 1970 [1]. Relational/SQL databases are schema-oriented, i.e., we need to define the schema of the data in advance. They are ideal for structured data and predictable workload. Their schema is largely rigid, complex and require expensive vertical scaling from time to time. In vertical scaling, we need to upgrade server in terms of increasing RAM, SSD, CPU, etc., in order to manage

the increasing load on the RDBMS. This approach is not scalable for handling Big Data which encompasses not just structured data, but also semi-structured and unstructured data. Not only this, the query workload is also unpredictable.

1.2.2 NoSQL Database

Not Only SQL (NoSQL) databases have been proposed as an alternative to SQL databases to handle the challenges posed by Big Data, specifically by unstructured nature of the data. NoSQL databases are characterised by flexible/dynamic schema and horizontal scaling, therefore, suitable to create a flexible structure based on the requirement. Commodity servers can be added in horizontal scaling as per requirement, thereby making NoSQL databases easily scalable. NoSQL represents a family of databases in which each database is quite different from others having literally nothing in common. The only commonality is that they use a data model with structure that is different from the traditional row-column relation model of RDBMSs. Graph, Document, Column-oriented, & Key-value pair are the four kinds of NoSQL databases.

The growth of unstructured data has led to interest in NoSQL databases. Nearly, 80% of the expected growth in data is in unstructured data as shown in Figure 1.1[1]. Social media platforms are the major source of unstructured data in the current times. Unstructured data is characterized by adhoc schema and therefore cannot be stored in SQL databases. NoSQL databases are much better suited due to their flexible schema. Text and multimedia are the most common forms of unstructured data. In addition to this, time series sensor data is also growing rapidly.

Despite the march of NoSQL databases in the past decade, approximately 60% of the world's current data is still in SQL databases (refer to Figure 1.2[2]). Almost 44% of the organizations use both of these kinds of databases for their data requirements (refer to Figure 1.3[2]). It is expected that the two database technologies will converge into a hybrid ecosystem (https://marketresearchmedia.com/nosql-market/). Already, 75% of the organizations are having both SQL & NoSQL databases in their portfolio (refer to Figure 1.4[2]). Presence of different types of NoSQL databases in a single organization is

[1]Source: IDC The Digital Universe. Dec. 2012

[2]Source: https://scalegrid.io/blog/2019-database-trends-sql-vs-nosql-top-databases-single-vs-multiple-database-use/

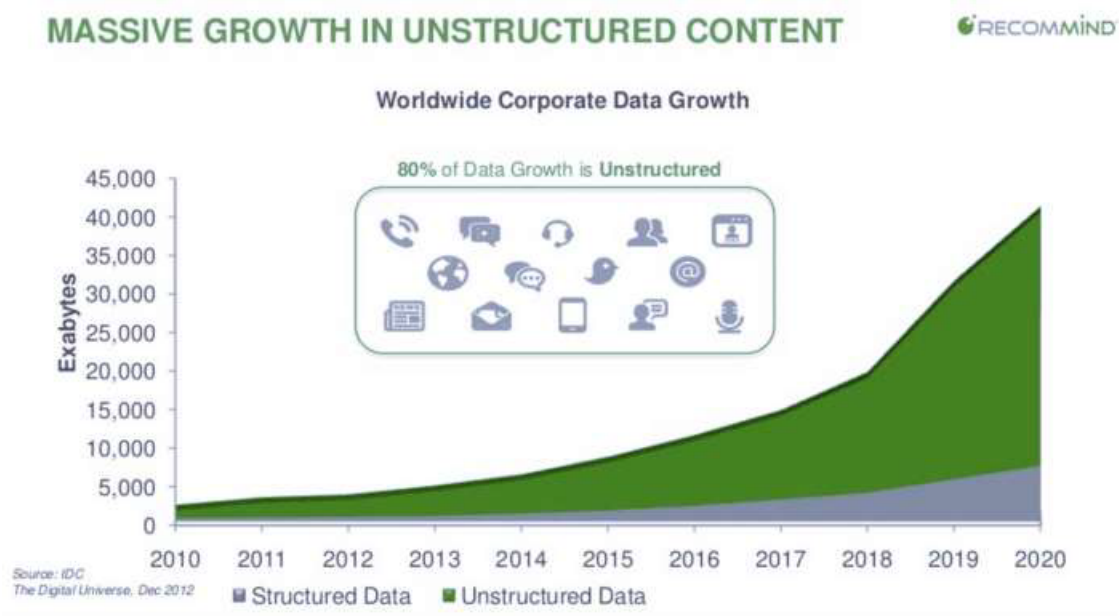

Figure 1.1: Massive Data Growth in past 10 years

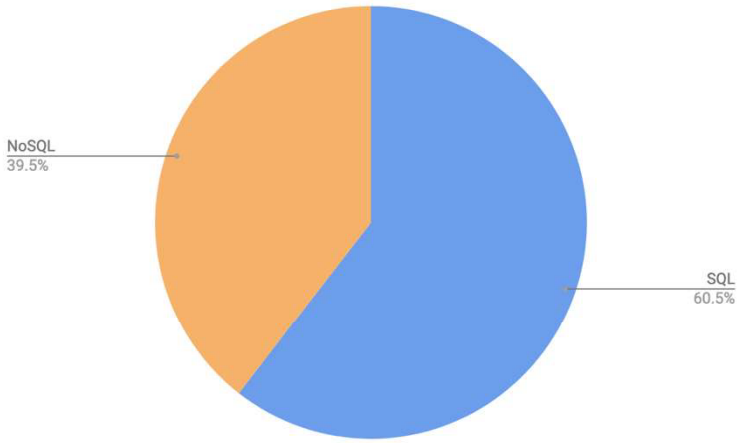

Figure 1.2: SQL vs. NoSQL Database Usage

also becoming a common trend.

Continued growth of data has also led to research about another aspect of data – ***Data Lineage/Provenance***. In very simple terms, it is about tracing the transformations data goes through from source to end-users. Data provenance has become critically important as we are living and will continue to live in a data-driven world. We are therefore interested in knowing as much as we can about the data based on which we are taking our decisions.

Veracity of big data that is defined as quality, accuracy and truthfulness of source

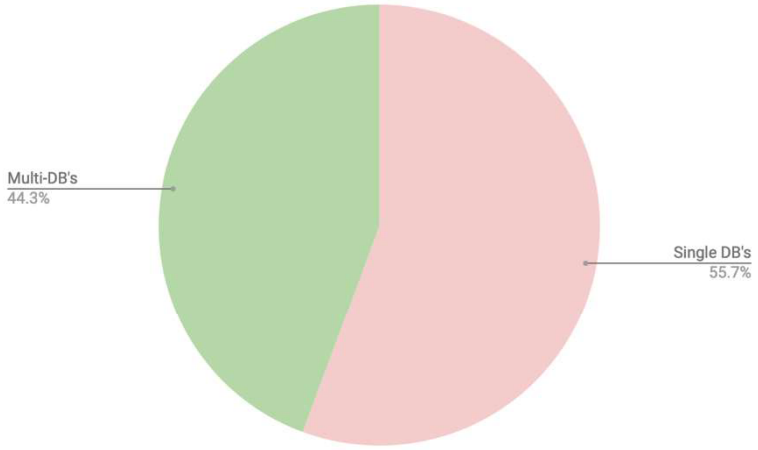

Figure 1.3: Single Database vs. Multi-Database Use

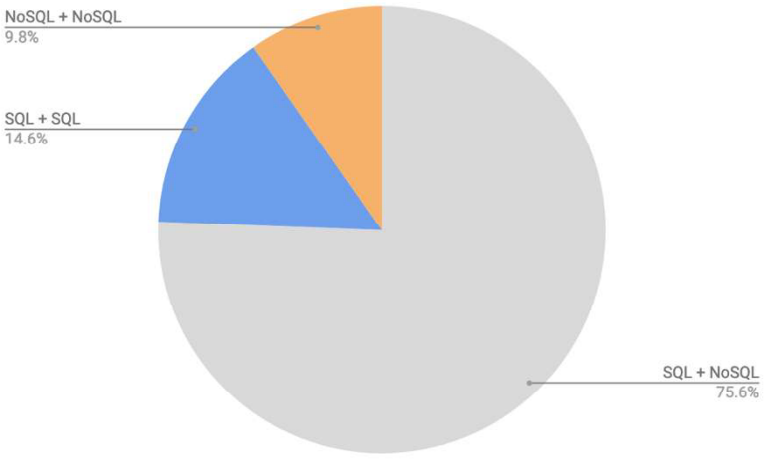

Figure 1.4: Multi-Database Combinations

of data is directly linked with data provenance. The provenance information for each query result can be determined and then visualized further to trace the direct and indirect sources of any data for different purposes such as auditing, error tracing, trustworthiness of source etc.

In the thesis, we attempt to design and develop data provenance frameworks for SQL, Graph, and Key-value pair databases which collectively store approximately 90% of world's data. The frameworks have been developed around the concept of a Zero Information Loss Database (ZILD) [3]. By a Zero Information Loss Database, we mean that no data value, no user, and no query and its result (seen at the time query was issued) is ever lost. ZILDs are very useful in tracking any "data manipulations" that have taken

place in an organization.

Our proposed ZILD in different databases aims to maintain all data update (insert, delete, and update) operations without any loss of information. It supports to perform historical data queries. Historical data query is defined as the query to know all updates/versions of a data object within any specific time range, or instance of a data object at any particular time. It also assists to capture provenance for historical queries. Historical query is defined as the query that was executed in the past and generates the same result (as seen in the past) in every subsequent execution.

1.3 Data Provenance

Data provenance, a kind of metadata information, describes the detailed, meaningful information of a data. Provenance information, can be like, what is the source of data, which tuples, nodes, attributes (i.e., fine-grained provenance) or relation/activity (coarse-grained provenance) is contributed to produce a result set, which transformation or a series of transformations (i.e., transformation provenance) are applied to generate the resultant data, who created/generated a data object in database, what are the previous versions of a data object if it is updated earlier and information about the execution environment (such as query statement, database state at the time of query execution etc) [9, 83, 13, 78].

The name "***Provenance***" mainly emanated from arts and humanities domain where it is used to know about source of artwork for determining its authenticity. But nowadays it is not limited to a specific domain rather commonly used in different domains also like astronomy, computer science etc., with its predetermined meaning. Its origin is the French verb '*provenir*', which means to come forth, originate; and a Latin word '*provenire*', from pro means '*forth*' + venire means '*come*'. The definition of provenance stated by Merriam-Webster Online Dictionary is "The history of ownership of a valued object or work of art or literature". As per Oxford Learning Dictionary, Provenance is "the place that something originally came from."

In computer science domain, provenance (also referred as *Lineage* [12], *Pedigree*) is mainly studied in two different perspectives viz., "*Workflow Framework*" and "*Database*

Management System" [30]. Provenance research issues and techniques in both the perspectives are different, i.e., provenance management technique in former one cannot be applied in later one and vice versa. Provenance in workflow frameworks, i.e., "*Workflow Provenance*" is a coarse-grained information that captures the information about process and entities involved in the process as a black box which allows framework re-execution. Whereas, provenance in database management systems named as "*Data Provenance*" captures the fine-grained information. Data provenance focuses on how any result is derived, what queries are executed, what operations are performed on data. In this thesis, our focus is on "*Data Provenance*".

Different definitions are given by different authors for Data Provenance. Simmhan et. al. defines data provenance as "A kind of metadata that pertains to the derivation history of a data product starting from its original sources" [30]. Buneman et. al. defines data provenance as "The description of the origins of data and the process by which it arrived in the database" [9, 13]. Lanter defined provenance (lineage) of derived products in GIS application domain as "The information that describes materials and transformations applied to derive the data" [12]. Greenwood et. al. further stated the provenance as "The metadata recording the process of experiment workflows, annotations, and notes about experiments" [21].

In line with these definitions, we propose the definition of data provenance in the context of database system as: "*Data Provenance is the information about direct and indirect sources of data which are contributing towards its generation along with the transformations applied on data, and the history of data*".

Provenance information can serve different purposes in a database system such as audit trail, knowledge rediscovery, incremental maintenance, rumour identification, fact investigation, justification of a query result etc [11, 12, 15, 30, 31, 48, 78, 147, 168]. In recent years, a piece of information published in an article on social media is also facing a critical challenge to determine its social provenance [156]. Provenance of a social data is termed as *social data provenance* or *social provenance* which involves following three dimensions viz., "*What*", "*Who*", and "*When*". *What* provides the descriptions about social media posts, *Who* describes the correlations among social media users, and *When* characterizes the evolution of users' behaviour over time. Like data provenance, social provenance also

describes the ownership and origin of such information. The continuously growing social media data is the major source of big data. This large volume of social media data is also termed as Big Social Data, and the provenance for Big Social Data is referred as *Big Social Data Provenance* [142]. It describes the origin, derivation and transformations of a social data element throughout its lifecycle. It is also categorized in following two categories based on its granularity level viz., *Fine-grained* and *Coarse-grained* provenance. In *Social Data Analytics*, the credibility of an analysis is generally depends upon the quality and truthiness[1] of input data which is assured by the *Social Data Provenance*. In this way, social data provenance plays a major role in clarifying opinions to avoid rumors [147], investigations [171] and explaining how and when this information is created and by whom. But distillation of provenance information from such a huge amount of complex data, however, is an extremely tedious task, due to its diverse formats. Not only the provenance generation, but efficient provenance querying [90] also like forward tracing (source to destination) upto any depth, backward tracing (destination to source) upto any depth, querying historical data etc., is much needed. Querying provenance [90] aid to solve different purposes such as error tracing, auditing, qualitative analysis, trustworthiness of derived data etc. as shown in Figure 1.5, and can be further applied to different applications domains such as bio-informatics, data diagnostics, information discovery etc.

1.4 Motivation

Efficient provenance capturing and querying framework in database systems plays a vital role in answering some decision making aspects such as "***What is the source of resultant data***"?, "***Why this data is generated***"?, "***How this data is derived***"?, "***Who has created this data***"?, "***History of data***"? etc. This information may be used in number of application domains such as Bio-Informatics, Data Analytics, Fact Investigations, Information Discovery etc. One of the real-life case study of provenance is in handling the food scandal in European food market in 2013 where provenance information was used to handle scandal due to wrong ingredient used in several processed foods by some manufacturers [149]. Thus, an efficient provenance framework is much useful for data processing. The various

[1]Dictionary meaning: The quality of seeming to be true according to one's intuition, opinion, or perception without regard to logic, factual evidence, or the like

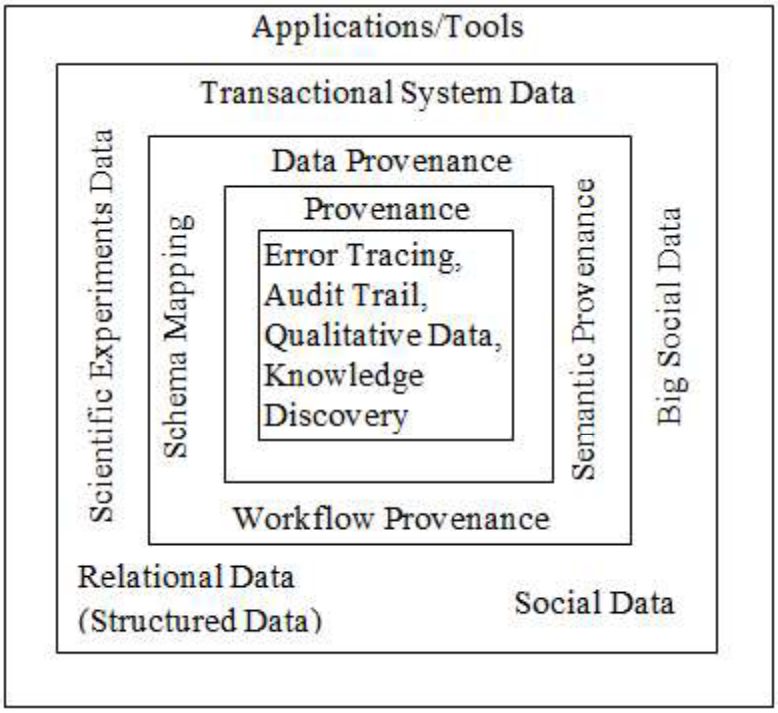

Figure 1.5: Overview of Provenance

issues in designing efficient provenance framework are as follows:

***Diverse Data Formats*:** In current scenario, data is generated from diverse sources in different formats such as structured, unstructured, and linked data but most of the existing provenance frameworks are focusing on one kind of database only. Further, the perspective of capturing, storing, and visualizing the provenance information are different in all the existing provenance frameworks. Thus, it is not feasible to provide a unified interface for different types of databases using existing provenance frameworks. Therefore, it would be useful to design the provenance frameworks for various SQL and NoSQL databases with a common approach using which a unified interface can be provided that can work seamlessly across different kinds of databases present in same organization.

***Time-aware Provenance Framework*:** For different applications such as audit trails, data diagnostics etc., "*Time-aware Provenance Frameworks*" is much valuable to maintain history of all updates without losing any information. Although, capturing provenance information about updates is very useful in querying historical data, tracing the source of derived data even after the source has been modified, but most of the existing frameworks have considered "*time*" as an optional parameter for provenance information. The

time-aware provenance framework may assist users with the ability to understand the analytical results and to repeat the analysis with different assumptions, parameters, or data sets, and to maintain complete history of data even if it is updated many times by anyone.

***Social Data Provenance*:** Today's Social Networking Sites (SNS) such as Twitter, Facebook, Instagram etc. are based on the tenets of mobilization and de-contextualization of information, where each user stands at the edge of a river of information to pick an independent data object for either repost or sharing with other users without including ownership and description of the content. It is difficult to know from where does this information has come, how it has been generated, or updated since its existence. In this sense, reliability of social data plays a major role in determining the quality [162], and credibility of an analysis. Several illegitimate activities are engendered by misusing these social content to accomplish various objectives. One of the main causes behind these illegitimate activities on social media is the separation of digital content from its provenance [138]. In Social Data Analytics, the credibility of an analysis generally depends upon the quality [172] and truthfulness of input data which can be assured by the *Social Data Provenance* or simply *Social Provenance* [117, 138, 152, 154, 153]. Most of the existing approaches in social media environment provide very limited provenance support as a coarse-grained provenance information.

Capturing provenance for big data applications is very challenging because of the high volume and unstructured data. A number of challenges are presented for provenance support in big data applications by different authors [103, 146, 128, 108] like automatic provenance capture, different granularity levels at which provenance need to be captured, provenance capturing overhead, and analyzing data via querying provenance etc. Distillation of provenance information from such a huge amount of complex data, however, is an extremely tedious task, due to its diverse formats. Therefore, the necessity to capture and query the provenance information about *Big Social Data (BSD)* has raised a growing interest in the era of social data analytics with several remarkable challenges. The rapidly growing large sized human-generated social data is termed as the Big Social Data. Provenance for Big Social Data is referred as *Big Social Data Provenance*. All of the existing provenance frameworks in big data environment are focusing on workflow provenance

and capturing coarse-grained provenance information only except one.

***Provenance Visualization*:** Along with provenance generation, provenance visualization support is also much needed. Different application domains such as error tracing, auditing etc., requires *Multi-Depth Provenance* queries to know about direct or indirect sources of any information. Although, a few existing systems support multi-depth provenance queries but are not efficient, as provenance is stored in relational database in such systems that deteriorate the query performance with varying traversal depth. Secondly, the issue of provenance querying has not been addressed much in an application and database independent way. Proposed languages in literature for querying provenance cannot be used for provenance query in other database systems. Thus, it would be useful to design a provenance query language that can support almost all possible type of provenance queries efficiently in application and database independent way.

The above issues motivate the development of time-aware provenance frameworks using *Zero-Information Loss Database* [3] concept for different database systems viz., *Relational Database*, *Graph Database*, and *Key-Value Pair (KVP) Database*. These frameworks can provide a unified interface which can work seamlessly across different databases. Proposed provenance frameworks capture provenance information for all queries executed on database including historical queries (i.e., queries executed in the past producing same results in every subsequent execution), provenance for data updates (i.e., insert, delete, and update queries). Provenance visualization support in frameworks such as forward tracing (i.e., querying provenance from source to destination), backward tracing (i.e., querying provenance from destination to source), and historical data queries (i.e., querying historical data for knowing all updates within any specific time range or instance of a value at any particular time) can be applicable to different applications domains such as bioinformatics, auditing, information discovery etc.

1.5 Research Foci

The foci of this thesis are to design the provenance frameworks that support to capture and query the provenance information for different type of databases. The frameworks should assist users with the ability to understand the analytical results generated by vi-

sualizing the provenance information and to maintain complete history of data even if it is updated many times by anyone. More specifically the research objectives of this thesis are:

RO1: Zero-Information Loss Database Design – Design Zero-Information Loss Databases (ZILDs) on top of different databases (i.e., Relational Database, Graph Database, Key-Value pair Database) which further support to develop time-aware provenance frameworks. ZILD supports data versioning to maintain history of all data updates as a provenance information. It also enables provenance capturing for historical queries.

RO2: Design and Development of Provenance Frameworks – Develop provenance frameworks on top of specialized Zero-Information Loss Databases designed for three different types of databases as mentioned in RO1. Provenance frameworks should capture provenance information for all queries executing on database and should also support Multi-Layer provenance generation. It should also support provenance generation for historical queries, and provenance for updates.

RO3: Provenance Visualization – Mere capturing provenance information is of no use, until it is queried/visualized efficiently. So, another objective is to provide provenance query support in all three databases for different purposes like justifying answers of a query statement, identifying error propagation, auditing, investigation etc., via different operations like forward tracking, backward tracking, and querying historical data for knowing all updates within any specific time range or instance of a value at any particular time (i.e., present or past). The frameworks should also support Multi-Depth provenance query with varying depth to know about direct or indirect sources contributed to a specific result, and how a specific source directly or indirectly contributing to any derived result.

RO4: Provenance Query Templates – Our another objective is to design various provenance query templates that may support to query provenance information in all possible ways independent of underlying databases and application. The main motivation behind proposing the templates is to facilitate users to pose useful & common provenance queries using the templates. These provenance query templates will guide the development of a full-fledged PQL in future.

1.6 Related Work on Data Provenance

Data management is the major concern in different application domains like bio-informatics, data discovery, social media, big data analytics etc., those are processing large amount of data collected from different sources/applications or generated from transformations applied on existing data. Applications are automatically generating metadata which describe the data like type and size of data, time of data creation etc., and maintaining the operation logs. Provenance information is also a kind of metadata that not only describe the data by adding some meaningful information to it [84], but also specifies the source of data [30], derivation history of data starting from its source, deriving process of data [13], and transformations applied on data [30]. It also captures the details about how data is updated over time [81]. It describes the complete production process of final product [149] which can be a digital entity or a physical entity. In [149], overview of provenance techniques with main focus on applications of provenance (What for?), types of provenance (What form?) and system requirements/resources for provenance (What from?) are presented.

Provenance information is essential to know about credibility of a piece of data, and for any decision making aspects. Using provenance information, one can determine the quality of data [9], trustworthiness [64] of data, explore direct/indirect sources of data, and reproduce [124] the experiments. Some other applications of capturing and visualizing provenance information are attribution, replication [149], collaboration, rumour detection [147], fact investigation [171], audit trails etc. Some real-life use cases of provenance [149] are 1) handling scandal due to wrong ingredient used in several processed foods by some manufacturers in European food market in 2013, 2) re-running of scientific experiments (ATLAS at CERN) in large volume of data by preserving both the data and procedure used to analyze them and relying provenance, 3) in complex data processing where pipeline comprise of cycle involving writing code, testing, analyzing result, refine code to have expected result, test again and so on till desired result is not obtained.

Provenance information is classified in four main types as a provenance hierarchy [149] containing 1) *provenance metadata*, 2) *information system provenance*, 3) *workflow provenance* and 4) *data provenance*. In this provenance hierarchy, provenance metadata is the most

general provenance type with little instrumentation required for capturing, whereas data provenance is at the most specific domain level requiring highest level of instrumentation to capture. From this hierarchy, *Workflow Provenance* and *Data Provenance* are mostly studied in literature. Workflow provenance systems are process-oriented [30] which generally capture provenance information at coarse-grained level like which process/activities, entities, transformations are involved in result set generation. In scientific domain, workflow provenance is very essential as it helps in reproducing the scientific experiments. Some of the workflow provenance systems are Taverna [25], Karma [72], Kepler [34], RAMP [100], HadoopProv [106] etc. On the other hand, data provenance is captured at fine-grained level, i.e., which sources like cell, tuple, attribute, relation etc. in relational database, nodes or links in graph databases, and keyspace, column family, rows in key-value pair database etc.) and how they are contributed to generate any query result.

Further, two main approaches used in existing provenance systems for capturing or representing provenance information are *Annotation Propagation*, and *Inverse Queries (Query Inversion)* [30]. In *Annotation Propagation* approach, annotations in the form of textual comments, labels, colors etc. are applied on data objects those are propagated from source to derived result during query execution. Provenance information is available in the form of annotations applied just after the query execution, thus this approach is also named as *Eager approach*. In Figure 1.6, some attribute values of relations PART and PARTSUPP are annotated with textual comments like Mouse is *Hardware* which can be *wired* or *wireless* etc. When query executes on these annotated relations, annotations applied on attributes are also propagated in result set. In the result set, we can see the Mouse supplied by Supplier S1 is annotated with Hardware as well as Wired, whereas it is annotated with Hardware as well as Wireless in case of Supplier S2. Depending on application requirements, these annotations are propagated which can be union or intersection of all.

On the other side, in *Query Inversion* approach, query/derivation-process/user-defined functions/methods are inverted to get source of any derived data. Query inversion approach is also named as *Lazy approach* because the provenance is computed whenever it is required, not during query execution itself as in annotation approach. For example, consider the PARTSUPP relation shown in Figure 1.6, display the total number of parts supplied by each supplier. The corresponding SQL query is:

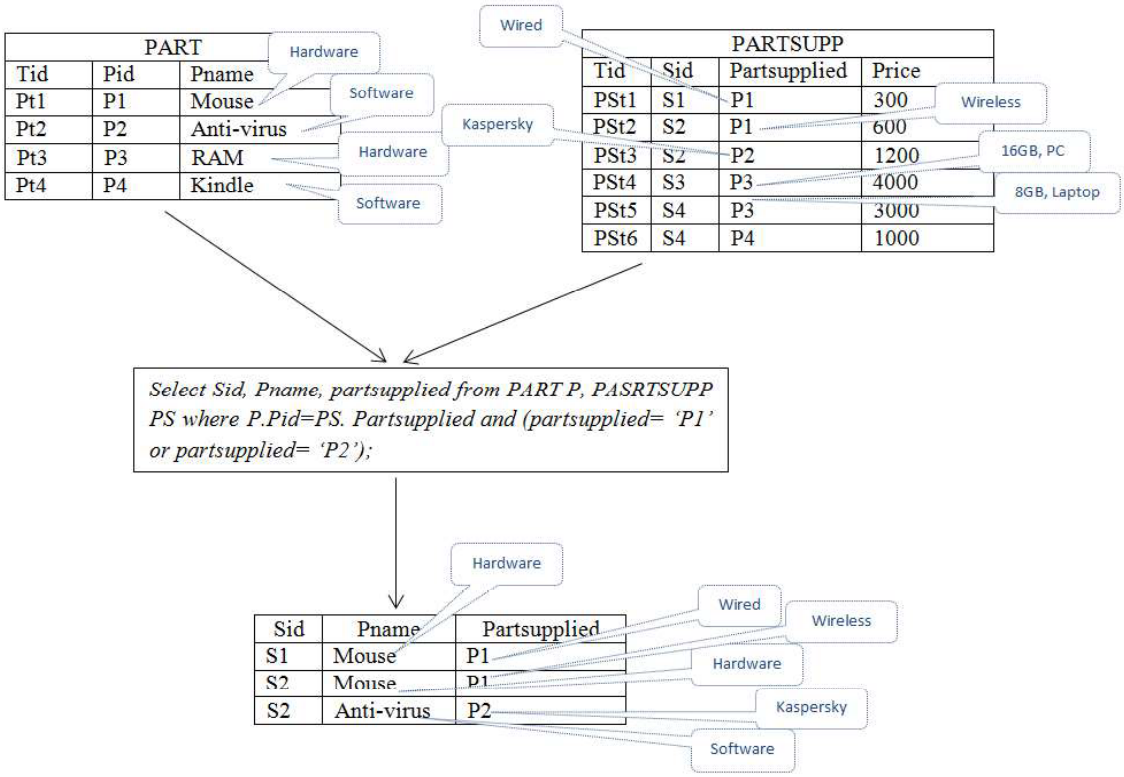

Figure 1.6: Annotation Propagation

SQL Query: select sid, count(partsuppname) from PARTSUPP group be sid;

Assuming some unexpected tuple appears in result. To analyse why it is appeared in result, inverse query is generated as below:

Inverse SQL Query: select * from PARTSUPP where sid in (select sid from PARTSUPP group by sid);

The above query will generate all the sources which are contributing to generate all results including unexpected result also. Although, query inversion approach represents provenance in a compact form just using a single inverse query/process/function for whole derived data, but all queries/process/functions are not invertible [30].

Data provenance for query results are further classified into three main categories viz., *Where-Provenance*, *Why-Provenance*, and *How-Provenance* [30, 149, 13].

Where-Provenance (Derivation basis [13]): Where-Provenance explains from *which location in database instance* any value in query result is copied, i.e., where does a piece of data comes from [13]. It means where-provenance refers to specific cells of a table in case of relational database systems [149]. In Figure 1.6, annotations applied on attribute values

are automatically propagated from source to result tuples whenever query executes. By examining the annotations on result tuple attributes, we can say that in result set attribute value *Mouse* in first and second tuples is copied from attribute *Pname* of first tuple with *Tuple-id (Tid) Pt1* of PART table. Attribute value *P1* in first result tuple is copied from attribute *Partsupplied* of first tuple with *Tuple-id (Tid) PSt1* of PARTSUPP table, whereas attribute value *P1* in second result tuple is copied from attribute *Partsupplied* of second tuple with *Tuple-id (Tid) PSt2* of PARTSUPP. Thus, in where-provenance, we can identify the source tuple attributes from where the values in result tuple attributes are copied using annotations on source and result tuple attributes. Although, where-provenance captures the source information "from where the value in result set is coming", but it cannot justify why any value is present in result tuple like first result tuple in result set in Figure 1.6 is present because of tuple Pt1 and Pst1 of PART and PARTSUPP tables respectively.

Why-Provenance (Wit or Witness Basis [13]): It identifies all the sources contributed towards generation of any query result, and explains why any result is present in result set. Why-provenance explains the justification of an existing result, thus capturing more useful information as compared to where-provenance. Given a query Q and a Relational Database Instance D, Q(D) generates all result tuples along with the set of source tuples $D' \subseteq D$ (as Why-Provenance) which are contributing towards result tuples generation. Set of source tuples, D' are sufficient enough for generating the same result set of Query Q executed on D' as in Q(D). For example, the query shown in Figure 1.6, the source tuples Pt1 and PSt1 are contributing towards first result tuple generation. Similarly, the source tuples Pt1 and PSt2, source tuples Pt2 and PSt3, are contributing towards second and third result tuple generation respectively. Table 1.1 shows Why-Provenance of each result tuple along with result tuples of example query given in Figure 1.6. Thus, we can say, why-provenance identifies the sources those contributed to produce the query result and are sufficient enough to produce the result by executing the same query again on provenance information only. Although, why-provenance justify the existence of a query result, but it does not provide any information about the derivation process of a result, i.e., how these sources in provenance information are contributing towards any result.

How-Provenance: It not only identifies the contributing sources towards any query

Sid	Pname	Partsupplied	Why-Provenance	How-Provenance
S1	Mouse	P1	<Pt1, PSt1>	<Pt1*PSt1>
S2	Mouse	P1	<Pt1, PSt2>	<Pt1*PSt2>
S2	Anti-Virus	P2	<Pt2, PSt3>	<Pt2*PSt3>

Table 1.1: Why-Provenance and How-Provenance for query in Figure 1.6

result generation as in why-provenance, but also explains about how these sources are contributing to derive the result. *How-provenance* is a superset of *where* and *why* Provenance. Semiring based provenance [46, 104] framework is proposed by Green et. al. which annotates tuples with elements from semiring to capture how-provenance which is further used in [97, 160]. When query executes, annotations for operators (like '+' for disjunction, OR and '*' for disjunction, join) are propagated along with annotation of tuples which finally annotate result tuple with provenance polynomial. Table 1.1 shows how-provenance as provenance polynomial like *<Pt1*PSt1>* for example query presented in Figure 1.6. It shows that the first result tuple is generated by joining Pt1 and PSt1 source tuples. Thus, how-provenance captures more meaningful information as compared to why-provenance, i.e., it not only captures the contributing sources as why-provenance does, but also how they contributed to derive the result.

Provenance Storage and Querying: Volume of provenance information can grow largely even more than the size of data with the increase in number of queries executed on database. Thus, there should be a systematic approach to store the captured provenance information so that system should provide efficient provenance queries. Few of the existing systems does not store provenance information [86, 112, 41], but provide provenance query support while capturing it. On the other side, some of the systems store complete provenance information in relational database for querying it later [31, 73, 28, 27], and a few store for specific use cases only but not for all [121]. Analysis of provenance storage and querying in relational and graph database is presented in [93, 123]. As provenance is a graph like structure, they found graph database more suitable for storing and querying provenance information especially for multi-depth provenance queries.

Open issues: A number of challenges and open problems are discussed in literature for provenance capturing and querying. One of the problems discussed in [81] is to build a general workbench for data analysis that should use provenance information for detecting and correcting errors by identifying the sources of errors, and then fixing them by propagating corrections. Another challenge discussed is to use eager/lazy approach or hybrid of these two for provenance generation. One of the major open challenges in data provenance is providing more versatile provenance query set [81] like querying updates along with justifying answers of queries, combining different type of provenance (where-provenance, how-provenance) which is beneficial in number of applications. For querying updates, there is a need to efficiently store all updates without losing any information, and query it methodically. Few of the existing provenance systems in relational database [31, 73, 121] supports historical data queries, but are not efficient for querying. As of now, none of the provenance frameworks in graph database and key-value pair database support historical data querying. Some of the challenges in big data environment are presented in [103, 108, 128, 139, 136, 146, 170] like tracing and storing provenance in a distributed heterogeneous environment, securing provenance etc.

Related work for provenance models in relational databases, graph databases, and key-value databases are presented in the following sub-sections.

1.6.1 Relational Databases

Allison Woodruff et. al. presented a fine-grained data lineage model [6] to support visualization in databases using inverse functions. They represented transformations as functions from one attribute domain to another. Groth presented a model for representing the user interactions with the system as a set of annotations [23], and present the whole activities as a DAG (Directed Acyclic Graph), i.e., a provenance graph. The users can then navigate forward or backward in the provenance graph to rediscover the information. Provenance in curated database [38] is presented by Buneman et. al. In the proposed model, the data collected from different sources is presented as a tree like data structure, and then copy-paste-database (CPDB) operations are performed on tree without deleting or updating any value in place, for generating different versions of data in database. A

query language is also proposed for querying the versioned data in generated tree. Later, they presented a use-case scenario of framework for archieving scientific data [22] by applying unique keys for every data object to identify different versions of data.

Some of the existing systems use annotation approach (shown in Figure 1.6) for provenance generation in relational database, in which annotations applied in different forms on source data are propagated from source to query result during its execution. One of the systems using annotation approach for provenance capture is DB-Notes [27, 28] in which zero or more textual annotations are applied on every attribute value. These annotations propagate from source to result set when query executes (i.e., Eager Approach) as "Where-Provenance". A Provenance Query Language, pSQL (an extended SQL version) is also developed to query data as well as provenance information. It provides limited support for annotation querying over single values only. Another annotation oriented provenance model MONDRIAN [41] is proposed by Floris Geerts et. al. It is an enhancement of DB-Notes, and is based on the concept of block as a group of values that span multiple attributes of one tuple. They assigned annotations on blocks in a form of colors, where different color specifies different annotations. It supports SPJU queries but with some limitations in the predicates of select and join queries. A query language, i.e., Color Query Language (CQL), based on Color Algebra is also proposed to support querying data as well as provenance in the model. The provenance model BDBMS [79, 62, 40] for maintaining biological databases is proposed by Eltabakh et. al. BDBMS is an extension of PostgreSQL, permit annotations on group of values that span multiple attributes of one tuple as in MONDRIAN as well as spanning multiple tuples also, but in predefined order on tuples in the relation. Separate annotation tables are created to store all annotations applied on each relation. So, during query processing both relations with applied annotations as well as corresponding annotation tables need to be specified in a query. Afterward, query rewrites by joining the tables specified in original query for propagating annotations along with result set which is a complex process. MMS [53, 54, 55] is an annotation based provenance model for metadata management by relating data with metadata using SQL queries as data values (referred as q-type values). The result of SQL query finds all attributes those are associated with specific annotation. But, the system does not provide support for automatic annotation propagation as in DB-Notes and

MONDRIAN. Although DB-Notes, MONDRIAN, BDBMS, and MMS provenance models use annotation approach to capture provenance, but it is a quite tedious process to assign annotations on each value or group of related values, and causing storage overhead also. In line with annotation propagation for provenance generation, Green et. al. proposed provenance semiring [46] for representing How-Provenance by applying annotation on tuples with values in semiring. Further, concept of relational query containment with annotation propagation is proposed in [20, 97].

Some methods are proposed in literature for provenance of updates and deletes [56, 15, 16]. Jennifer Widom et. al. [31, 35, 33, 74, 26, 36, 43, 44, 49, 51, 92, 50, 68, 66, 73, 67, 44] proposed the Trio system that support provenance and Uncertainty Lineage Database (ULDB). It is the first system to manage data, its accuracy, and lineage for uncertain relational databases using query inversion approach. They defined the lineage on materialized view in a data warehouse as the maximal set of tuples from source tables those contributed to it. They interpreted the view as a query tree in which input tables and algebra operators are represented as leaf nodes and internal nodes of tree respectively. Query tree is evaluated in a bottom-up manner where the algebra operators accept the result set from its child nodes, and generate the result set for its parent node. Provenance is generated for ASPJ (Aggregate,Select,Project,Join) views by inverse queries in Trio. Inverse queries generate the provenance for each tuple that is further stored in a special lineage table. Lineage table includes following attributes, i.e., tupleID, derivation-type, time, how derived, lineage-data etc. This model is suitable for capturing historical lineage from the expired portion of database, i.e., the data which is not valid now. They also implemented a query language called TriQL(Trio Query Language) which is based on SQL to support provenance and uncertainty in databases. But it has high storage overhead as it stores complete lineage information for every tuple in lineage table. In WHIPS model [5], provenance of views in data warehouse using query inversion approach is captured lazily (on-demand) as compared to eager approach (automatic during query execution) in annotation propagation. Provenance is captured as per Lineage-CS semantics [7, 11, 12, 10, 14]. The proposed algorithm is suitable to capture lineage for set-ASPJ queries. For every provenance computation, model requires execution of various SQL queries and user-defined functions which increases the complexity of system and deteriorates its per-

formance. Another issue is the relation between result tuples and their provenance are not preserved.

Glavic et. al. proposed Provenance Extension of the Relational Model (PERM) [80, 86, 112]. They identified Perm Influence Contribution Semantics (PI-CS) based on why-provenance model. Perm generates the provenance for ASPJ (Aggregate, Select, Project, Join) queries, and is suitable for both set-semantics and bag-semantics queries. It is built upon PostgreSQL and capture provenance information on demand. It uses query inversion approach [30] to capture why-provenance. Result of query inversion contains actual result set including provenance information as annotations in separate fields. Perm uses PostgreSQL properties for querying, storing and query optimization. It also uses an SQL language extension called SQL-PLE to enable a user to issue provenance queries while capturing but does not store provenance information at all. Bahareh et. al. [121] proposed GProM (Generic Database Provenance Middleware) provenance model, which uses query inversion approach to capture provenance information. This model is capable to capture transaction and historical provenance using audit log. Although, it requires a very complex query rewriting to capture the provenance for past queries using audit log, yet it may not capture all the information in audit log. It does not store provenance information explicitly for further querying. The model stores the provenance information for a few use cases only where provenance reconstruction is not possible from audit log. Glavic et. al. further developed a debugging tool TRAMP (i.e., TRAnsformation Mapping Provenance) [87] implemented over PERM. TRAMP includes all the features of PERM model, and also traces schema mapping (i.e., Mapping Provenance) and transformation queries (i.e., Transformation Provenance).

Under schema mapping provenance, Green et. al. presented an ORCHESTRA [29, 47, 64, 48, 88] data model. It is developed as per the need of collaborative data sharing between different users. It can efficiently manage the exchange of updates performed by any user in a group (i.e., peer-to-peer network) using schema mapping, and captures how-provenance to describe the updates which are represented as functions in the semiring provenance model [97, 46, 104]. The model also supports trust management by annotating result tuple generated after every transformation with a trust level of a user. Users can annotate each tuple and transformation with T (to Trust) or D (Not to

Trust). As updates are exchanged with other users, it gets filtered based on trust conditions that uses the provenance data in updates. A provenance query language ProQL [90], is also proposed for provenance graph, based on provenance semiring. This model is suitable for small dataset only because of high communication overhead with frequent update exchanges. Another system for provenance in data exchange [32] is proposed by Velegrakis et. al. Provenance information for mapping derived data from source data is generated along with the transformation applied to derive the data. Annotations on data values are applied to associate data values with their meta-data information which can be further queried along with data values. An extended query language, i.e., Meta-data extended Query Language (MXQL) is proposed for querying provenance and mapping information.

Few workflow systems also provide support for provenance (i.e., workflow provenance) capturing and querying. Workflow provenance systems generally capture coarse-grained provenance information such as which processes/activities, entities, transformations involved in result set generation, but not the detailed fine-grained provenance information as in case of data provenance. One of the workflow provenance system, Taverna [21, 18, 19, 17, 24, 25] is developed under myGrid for capturing provenance for e-science experiments. It represents provenance as RDF triplets. Another provenance management system Karma for Data-Driven workflows is proposed with three different versions, i.e., Karma 1 [72], Karma 2 [71], and Karma 3 [77]. In this, workflow comprises of different services which can generate their provenance if participating in any activity, and store it further in relational database. A generic provenance framework for popular scientific workflow management system, Kepler [34, 37, 75, 76, 65, 58, 82, 42] is proposed for capturing provenance for result data as well as processes involved in experiment via Provenance Recorder (PR). It also enables efficient rerun of a workflow by querying stored provenance data using Smart Rerun Manager (SRM). Anand et. al. [75, 76] introduced fine-grained provenance generation for COMAD (Collection-oriented modelling and design) developed in Kepler workflow which is not supported in other workflow systems. It also supports query language for querying provenance based on XPath. VisTrails [63, 52, 60, 70, 69, 61, 39], another workflow management system captures the history and

provenance of workflow execution which is further stored in Relational or XML database. Correspondingly, vtPQL (VisTrail Provenance Query Language) query language is proposed for querying provenance of execution and different versions.

Vicknair et. al. [93] suggested the need of graph database for storing and querying data provenance. They analyzed the performance of MySQL and Neo4j from a data provenance perspective. They used DAGs (Directed Acyclic Graph) to store provenance information. They reported that querying on character data is efficient in Neo4j as compared to MySQL. Further, they reported that the performance of provenance queries in MySQL deteriorated with increase in provenance data size. Kirby et. al. [123] compared the performance of Neo4j and MariaDB for synthetic pedigree data set generated by population generator. Dataset includes persons (birth date, death date) and their marriages. Related persons and marriages are linked with each other. This linkage is a kind of provenance information. They analyzed both the databases in terms of scalability, query execution time, and ease of query expression.

1.6.2 Graph Databases

Social media is a group of web based applications build on technological and ideological foundations of Web 2.0, and Social Networking Sites (SNS) are applications through which users can create and exchange the information with others including short messages, blogs, images, multimedia files etc [89]. Among all social media networks, Twitter has become one of the most popular social networking/micro-blogging sites, allowing users to share their thoughts with massive audience. Its popularity as a huge source of information has led to research in various domains. Researchers and practitioners can obtain this information from twitter through public APIs without any cost [99].

The term data model is widely used in the field of data science and analytics. The definition of data model is defined in [4] as a set of conceptual tools used to model real world entities and their relationships. In line with the data model concepts and ever increasing technological advancements, a time-varying social network data [107] is proposed, in order to preserve temporal information using Neo4j (NoSQL graph database) [157, 134]. In Neo4j, the collected data are represented by a property graph where nodes

are used to represent individual entities, while edges represent the interactions among them, and nodes and edges may have their own properties in the form of key-value pair. Cypher query language (CQL) is used to analyze a huge social network datasets, imported from twitter network [154, 171]. These research works explores the suitability of Neo4j graph database for efficient storage and to perform fast data analytics [165, 57, 155], and recommendation in social network datasets.

In social data analytics, the accountability of an analysis is largely relying on the data quality and trustworthiness of input data. Recently, social data provenance [156] has gained a lot of attentions, as this helps in identifying sources of a given piece of information propagating through a social media network. Baeth et. al. in [144, 142] identified the significance of social data provenance [156] for various applications like tracking source of data, assessing data quality, risk-awareness [102], mapping trustworthiness of data, data management, and big data analytics [136] etc. Further, they also explored the effectiveness of a standalone centralized social provenance system as per scalability and responsiveness to manage a large sized social provenance data. In this way, Social Feed Manager (SFM), an open source tool, is proposed in [167]. SFM is developed to capture only metadata information of a tweet as provenance information that is inadequate for an effective social provenance framework. Social provenance [117] can be integrated with social computing system [152] to support transparency in data propagation, updates etc. A theoretical framework to realize a provenance model for social media with the capability to capture sufficient amount of provenance data in social media by using social media information itself is proposed in [94].

In this way, social data provenance is playing a vital importance in various data related activities including data management, database validation, big data analytics [103, 136] etc. Several existing systems which support social computation [152] are suffering from lack of transparency which can be addressed by integrating social data provenance in such systems [117]. From last few years, social media has become one of the most growing communication medium for peoples. Social provenance associated with social media can help in fake claims, fact checking, dispel rumors and to clarify opinions. However, neither social media nor application developers explicitly provides such provenance information to the recipient users to verify the credibility of a statement appeared in the

social media. Determining and tracking provenance data for a social media environment is a challenging task as social media is a decentralized and dynamic environment. In literature, mostly two provenance models viz., open provenance model (OPM) and World Wide Web Consortium (W3C) PROV model are used for provenance management in social media.

Social data provenance in graph database model: Ranganath et. al. [114, 118] developed a web based tool that captures information about pre-defined provenance attributes such as name, gender, religion, location etc., from different social accounts associated with a particular twitter user. Although, these provenance attributes capture complete details of a social media user, but it neither provides a provenance path nor a propagation history and updates of a piece of information published on a social media platform. Further, a provenance path algorithm [113] is proposed to capture provenance paths of an information, to explain how this information propagates in a social network. In [110], an approach based on provenance graph is proposed to identify the malicious node in a distributed network. Provenance graphs are generated by applying some derivation rules on provenance logs of network. These graphs are further analyzed for discovering high compression substructures to identify common execution pattern and malicious nodes as well. The proposed method is deemed fit for a static network but not for a dynamic network where provenance graphs would constantly change during execution. Secondly, the logs have an ad-hoc structure, not readily available for effectively querying and may not capture complete provenance information. To reconstruct and integrate provenance of messages in social media, a workflow provenance model PROV-SAID [129, 135, 163] based on W3C PROV data model is proposed. Although, the proposed solution identifies the posted tweets those are copied from other published tweets without giving credit to original tweeter like a retweet, but it is suitable for a small dataset only. The work of [143] introduced a topic-focused trust model to assess trustworthiness of both tweet and the user who posted that tweet related to a specific event in Twitter's network. This model first extracts trustworthiness features from news articles on a given topic, and then rank the tweets with a trust level based on similarities found in previous tweets on same topic. Afterward, trust further propagates to evaluate trustworthiness of that tweet.

A python library toolkit [130] is proposed to capture provenance for workflows that

collect provenance about every process in the workflow, but not suitable to capture provenance at fine-grained level, i.e., how an element has been generated in the result set etc. A provenance framework based on algebraic structure of semirings, for three specific graph algorithms is presented in [160], to compute provenance of regular path queries (RPQ) over graph database via applying annotations like labels and weight functions which is a quite complex process. A provenance model for vertex centric graph computation and a declarative data-log based query language is presented in [169], to capture and query graph analytics provenance for both online and offline mode. In some other way, a Q-Chase based algorithm is introduced in [168] for efficient implementation of a query chasing process and to compute query rewrite.

1.6.3 Key-Value Pair (KVP) Databases

Social data analytics is a research field that integrates social communications with data analytics. It extracts meaningful insight from extensively large data sets. It can be used to understand the user's behavior, and to model social interactions among social media users. *Big Social Data* [161] is mainly characterizes by Volume, Velocity, Variety, Veracity where volume means rapidly growing social data, velocity is related to the dissemination of data with tremendous speed, variety refers to diverse formats of social data, and veracity refers to quality, accuracy and truthfulness of source of data. Veracity of big data is directly linked with data provenance. The volume, velocity, and variety of Big Social Data further introduced the challenges of capturing provenance [138] and evaluating trustworthiness [172] of social data [89].

Several research works are carried out to identify the suitability of NoSQL database to manage big social data with efficient storage, fast querying, and horizontal scalability [85, 91, 105, 141]. Wang et. al. [105] describe the basic principles of NoSQL databases such as CAP, BASE, and Eventual Consistency theorems as the foundation stones. It revealed some facts about Apache Cassandra as a part of some popular social networking sites such as Twitter, Facebook etc., and demonstrated an online trading system design based on Cassandra. An architectural overview of Apache Cassandra [85] is presented to explore the suitability of Cassandra for efficient storage and querying on high volume of airline

flight's distributed database. It's a distributed storage system used to manage a high volume of structured data spread across several nodes, and provides high performance, scalability, wide applicability, and high update throughput with low latency [91].

Different approaches are proposed to model a huge volume of Twitter data set in Apache Cassandra NoSQL database for an efficient querying [125, 127, 131, 159, 141]. Due to ever increasing technological advancements, NoSQL databases have become the favourable choice of several application developers for schema design in big data era, still schema design/data modeling for a column store is a challenging task that depends on the data model of that application, and a set of queries on that data. To address this challenge, a systematic approach on database schema design in NoSQL column store is presented in [125] that rated and ordered a schema design based on a scoring function. Another query driven big data model design for Apache Cassandra [127] is proposed using data nesting and data duplication. The proposed model also defined mapping rules and patterns for logical data models design in Cassandra by demonstrating a data modeling tool that automates entire modeling process. Another query driven data model design for Cassandra is proposed in [131] that parses and analyze each query before its execution, and if any query language constructs related to schema modification such as CREATE, ALTER, DROP etc. is found, then the metadata is updated with new information. A query driven data model is proposed in [159] for twitter dataset to perform efficient read and write operations on massively stored tweets, and to maintain user's timeline in Apache Cassandra. Because of Cassandra's high performance and availability, its future scope in implementing relationships among twitter users and their friends and followers are also suggested. To explore the suitability of NoSQL databases in storing massive amount of discrete time series datasets, few data modeling schemas are presented in [141] that emphasis on sequential data storage and flexible schema design in Cassandra and MongoDB respectively. In line with NoSQL database, a performance comparison between relational and NoSQL database are presented in [140] that investigate the performance of NoSQL database in a scalable distributed environment in terms of processing speed, query language support, fault tolerance, and flexible schema design.

The importance of social data provenance in social media is also presented in [156, 117, 152] with several key challenges such as measuring quality and truthiness of social

data, provenance storage, provenance querying etc [146, 108, 128, 139, 103, 170, 136]. Boris Glavic [103] identified an essential requirement of Big Provenance in the field of Big Data Analytics for auditing, debugging, transforming, modeling and evaluating quality and trust in big data. He examined that big data benchmarking can be used to measure system's capabilities, performance bottleneck and metrics. He also stated that querying and storing provenance in a distributed heterogeneous environment is a big challenge in big data provenance research. Dunren et. al. [108] presented an overview of various platforms and frameworks used for big data processing, mining and management. They addressed several challenges in the era of big data research including provenance tracing which directly contributed to trustworthiness and accuracy of source data and derived data. Alfredo Cuzzocrea [128, 139] identified some major challenges and issues in the era of big data provenance research including provenance data model design for a heterogeneous environment, and to develop provenance querying and visualization tool. Wang et. al. [136] identified that the huge volume, variety and veracity of big data have become the obstructions in big provenance design. Reference architecture for big data provenance in workflows is also presented to address these challenges and opportunities. The work of [146, 170] focuses on current methods and approaches in big data provenance research and suggested a need of further research in all aspects of big data provenance including provenance storage, recording, querying, securing etc.

Provenance in Key-Value Pair System: A provenance data model for data intensive workflows is proposed in [95] to capture provenance information for Map Reduce workflows using Kepler-Hadoop framework. The proposed provenance model is a good initiation for scientific workflows; however it is not much efficient in terms of storage space and query execution overhead. In line with the provenance data model for scientific workflows, RAMP model is proposed in [98, 100] for Generalized Map and Reduce Workflows (GMRWs) using a wrapper based approach for provenance capturing and tracing. In this model all the transformations are either map or reduce functions rather than having one map function followed by one reduce function. Next, HadoopProv model [106] is introduced for provenance tracking in Map Reduce workflows, where provenance tracking takes place in Map and Reduce phases separately and construction of provenance graph is deferred at query stage, to minimize the temporal overhead. Applications of

standard PROV-DM model is proposed in [150] to manage provenance data for bioinformatics workflows in a cloud computing environment using different families of NoSQL databases.

To satisfy the need of Big Data Provenance, a rule based framework for provenance identification and collection from log files is proposed in [111]. Although, the proposed framework reduces the source code instrumentation yet raises several questions about completeness of provenance information, as logs may not capture complete information including derivation process. Another big provenance framework is proposed in [109] for provenance collection and storage in an unstructured or semi-structured format, for scientific applications. The proposed framework is light weighted and built on multi layered provenance architecture that supports a wide range of provenance queries. A provenance model for Apache Cassandra, i.e., a key-value pair database, is proposed in [115, 116] to capture provenance information using provenance policies. In this model, provenance querying is performed through resource expressions and a set of predefined operators. The proposed model is implemented on a small sized patient information system and uses legacy thrift APIs rather than CQL3 that makes it difficult to write a query. Various change data capture (CDC) schemes are investigated in [133] for Apache Cassandra to track modifications in source data. The logic of each scheme is implemented in Cassandra by combining a Map Reduce framework with distributed computing. A layer based architecture for provenance collection and querying in scientific applications is presented in [120], which stores semi-structured provenance documents in MongoDB in a BSON format. The proposed architecture is prominent for simple queries but not efficiently respond to complex queries.

1.7 Research Gaps

From the available literature on provenance frameworks in relational databases, it is evident that most of the existing data provenance frameworks either do not store the provenance information at all (support querying provenance as it is generated), and even if they store the provenance information, they store it in relational database for querying and suffers from query performance issues. Another issue is Multi-Layer Provenance capture

and Multi-Depth Provenance query. Tracking how a particular piece of data is derived directly or indirectly becomes very challenging that requires provenance capturing at various stages of data life cycle. This results in the need of *Multi-Layer Provenance Capture* and efficient *Multi-Depth Provenance Querying*. Few of the existing provenance frameworks such as TRIO and ORCHESTRA supports multi-depth provenance queries but store the captured provenance in relational database only. Although, relational databases can store graph data such as provenance graph, but they are not efficient for multi-depth querying on provenance. Because, with increase in data queries, the volume of provenance information also increases, leading to increase in number of join operations for provenance queries with varying depth. As discussed earlier, provenance for historical queries are also very much needed for applications such as auditing and error tracing, but only TRIO [31, 73] and GProM [121] support provenance for historical queries.

It is obvious from available literature that most of the existing approaches for provenance in graph databases are not scalable to track provenance metadata for social media efficiently. Existing frameworks primarily focused on workflow provenance that captures coarse-grained provenance only rather than detailed fine-grained provenance information. Although, a few frameworks support data provenance that capture fine-grained provenance information, but they do not support all type of queries for provenance capture. As per our knowledge, none of the existing frameworks support provenance for historical queries and provenance for data updates.

Most of the existing provenance frameworks for big data environment are suitable to capture provenance for workflows that do not capture detailed provenance information. Secondly, some of them are not suitable to capture provenance information for a large sized data set including all types of query sets.

In the last, issue of provenance querying has not been addressed much in an application and database independent way. Proposed languages in literature for querying provenance cannot be used for provenance query in other database systems. Thus, it would be useful to provide common provenance templates to design a provenance query language that can support almost all possible type of provenance queries efficiently in application and database independent way.

1.8 Research Contributions

This thesis makes number of contributions to provenance frameworks in different databases. Following are the main contributions of this research work:

1.8.1 Zero-Information Loss Database Design

We first propose zero-information loss database design for different databases as follows:

1.8.1.1 Relational Database

ZILRDB (**Z**ero-**I**nformation **L**oss **R**elational **D**atabase) is developed on top of conventional relational database using object-relational database concepts like user-defined data types and nested tables to maintain all updates efficiently without any loss of information. ZILRDB supports data versioning to maintain history of all data updates as provenance information that further supports in querying historical data (i.e., knowing all updates on data within any specific time range or instance of a value at any particular time). It also enables provenance generation for historical queries (i.e., queries executed in the past producing same results in every subsequent executions and capturing provenance information for the same).

1.8.1.2 Graph Database

ZILGDB (**Z**ero-**I**nformation **L**oss **G**raph **D**atabase) is implemented on top of Neo4j Graph Database that supports data versioning to maintain history of all data updates (i.e., insert, update, and delete) as provenance information. It also enables provenance for historical queries in graph database along with querying historical data.

1.8.1.3 Key-Value Pair (KVP) Database

ZILKVD (**Z**ero-**I**nformation **L**oss **K**ey-**V**alue Pair **D**atabase) is designed on top of Key-Value Pair (Cassandra) database that maintains information about all data updates (i.e., insert, update, and delete) without any information loss as provenance information. It also supports provenance for historical queries along with querying historical data.

1.8.2 Design and Development of Provenance Framework

1.8.2.1 Relational Database

We designed and implemented DPHQ (**D**ata **P**rovenance for **H**istorical **Q**ueries) framework on top of ZILRDB to capture provenance for all queries including historical queries, provenance for data updates etc. Qualitative analysis of existing provenance solutions and our proposed DPHQ framework based on an evaluation matrix including type and level of provenance generation, type of queries supported for provenance generation, provenance storage, and provenance visualization support is shown in Table 1.2. Main contributions for this work are:

1. ***Provenance Relational Algebra (PRA)*:** Provenance Relational Algebra (PRA) which is an extension of traditional Relational Algebra is proposed for capturing provenance information for all queries executing on ZILRDB. PRA supports provenance for ASPJU (Aggregate, Select, Project, Join, Union) queries.

2. ***Multi-Layer Provenance Capture and Multi-Depth Provenance Query*:** DPHQ supports Multi-Layer provenance capture for generating complete provenance information. That means, it not only captures direct contributions towards result tuple generation but indirect contributions also. Framework also supports efficient Multi-Depth provenance queries by storing provenance of relational queries in Neo4j Graph Database also.

3. ***Querying Current State*:** Proposed framework allows querying the current data version and captures the provenance information for the same. The captured provenance is stored in both relational as well as graph database for further analysis.

4. ***Querying Historical Data*:** It enables querying historical data by querying nested tables using extended SQL constructs viz., "instance", "all", "validon now", and "validon 'date'".

5. ***Provenance for Historical Queries*:** Proposed framework supports to generate provenance for historical queries.

6. ***Provenance Visualization*:** Captured provenance can be easily visualized in both

Provenance Framework	Provenance Representation	Subject Of Provenance	Provenance Capturing Support	Provenance Storage	Provenance Dissemination
DBNotes [28]	Data Provenance with Annotation Propagation	SPJU (Select Project Join Union) Queries	Suitable for Currently Executing Query	Complete Result Table with Annotation	SQL/PSQL Queries, Visual Interface
MONDRIAN [41]	Data Provenance with Annotation Propagation	SPJU (Select Project Join Union) Queries	Suitable for Currently Executing Query	No Provenance Store	CQL (Color Query Language)
TRIO [31]	Data Provenance with Query Inversion, Historical Data Provenance	ASPJ (Aggeragte Select Project Join) Queries	Suitable for Currently Executing Query and Historical Queries	Complete Lineage Table	SQL/TriQL Queries, Support Multi-Depth Provenance Query
PERM [86]	Data Provenance with Query Inversion	ASPJU (Aggeragte Select Project Join Union) Queries	Suitable for Currently Executing Query	No Provenance Store	SQL-PLE, PERM Visual Browser
TRAMP [87]	Data Provenance with Query Inversion, Schema Provenance using Annotation, Transformation Provenance using query rewrite	ASPJU (Aggeragte Select Project Join Union) Queries	Suitable for Currently Executing Query	Schema of query extended with provenance attribute to store transformation provenance as annotation function	SQL-PLE, PERM Visual Browser
GProM [121]	Data Provenance with Query Inversion, and Annotation Propagation with Audit Logs, Historical Data Provenance	ASPJU (Aggregate Select Project Join Union) Queries	Suitable for Currently Executing Query and Historical Queries	No Explicit Provenance Store	SQL Queries
ORCHESTRA [64]	Schema Provenance	SPJU (Select Project Join Union) Queries	Suitable for Currently Executing Query	Provenance Table per Mapping including Data Instance	Datalog Rules in RDBMS, Support Multi-Depth Provenance Query
Our Proposed Framework (DPHQ)	Data Provenance with Query Inversion, Transformation Provenance, Historical Data Provenance	ASPJU (Agregate Select Project Join Union), Insert, Update, and Delete Queries	Suitable for Currently Executing Query and Historical Queries, Multi-Layer Provenance Capture	Provenance Polynomial and Query Information in both relational and graph database	SQL Queries/ Cypher Queries, Support more versatile querying on provenance, Support Multi-Depth Provenance Query

Table 1.2: Qualitative analysis of different provenance solutions for Relational Database with proposed DPHQ Framework

relational and graph database, i.e., how it is represented, what is provenance for each result tuple of every query executed.

1.8.2.2 Graph Database

We designed and implemented SDP (**S**ocial **D**ata **P**rovenance) framework on top of ZIL-GDB which efficiently captures provenance of all queries including historical queries in social media data, and generates a provenance graph database for further querying. Qualitative analysis of existing provenance solutions and our proposed SDP framework based on an evaluation matrix including level of provenance granularity, type of queries supported for provenance generation, provenance visualization, and its applicability is shown in Table 1.3. Main contributions of this work are:

1. ***Modelling Social Media (Twitter) Data in Neo4j Graph Database*:** We propose to model twitter data from CSV files in Neo4j Graph Database using timeline approach for efficient query execution.

2. ***Multi-Layer Provenance Capture and Multi-Depth Provenance Query*:** Proposing an algorithm to generate provenance information for a query set including select queries, aggregate queries, and historical queries, and to store it in **P**rovenance **G**raph **D**atabase (PGDB). Framework also supports Multi-Layer provenance capture and efficient Multi-Depth provenance query as in DPHQ framework in relational database.

3. ***Querying Current State*:** Proposed framework allows querying the current data version and captures the provenance for the same. The captured provenance is stored in Provenance Graph Database for further analysis.

4. ***Querying Historical Data*:** It enables querying historical data by extending Neo4j Cypher Query with following constructs viz., "instance", "all", "valid_on now", and "valid_on 'date'".

5. ***Query through Time*:** It also supports for query data any time in the past as well as some time range as predicate specified in query statement.

Provenance Model	Provenance Granularity	Provenance Capture						Provenance Visualization			Application Domain
		Select Query	Aggregate Query	Historical Query	Update Query	Insert Query	Delete Query	Provenance Querying	Justifying Query Results	Historical Data	
SFM [167]	Fine-Grained Level (Only Metadata information of tweet)	Yes	No	No	No	No	No	Yes	Yes (Limited)	No	Generic
Web-Based Tool [114, 118]	Fine-Grained Level (Pre-defined attributes of social media user)	Yes	No	No	No	No	No	Yes	Yes (Limited)	No	Generic
Seeking Provenance Paths [113]	Fine-Grained Level	Yes	No	No	No	No	No	Yes	Yes	No	Generic
Substructure Mining [110]	Log Based	Yes	No	No	No	No	No	Yes	No	No	Application Specific (Identify misbehaving node in network)
PROV-SAID [129, 135, 163]	Workflow Level	Yes	No	No	No	No	No	Yes	Yes	No	Application Specific
Semiring Provenance [160]	Fine-Grained Level	Yes	No	No	No	No	No	Yes	Yes	No	Limited for three specific graph algorithms
Ariadane [169]	Fine-Grained Level	Yes	No	No	No	No	No	Yes (Online and Offline)	Yes	No	Vertex Centric Graph Analytics
Our Proposed Framework (Social Data Provenance (SDP))	Fine-Grained Level	Yes	Yes	Yes	Yes	Yes	Yes	Yes	Yes	Yes	Generic (use case in terrorist attack investigation)

Table 1.3: Qualitative analysis of different provenance solutions for Graph Database with proposed SDP Framework

6. ***Provenance for Historical Queries*:** Proposed framework enables to trace the provenance of each query result of a query executed in the past, i.e., historical query.

7. ***Provenance Visualization*:** Captured provenance can be easily visualized in graph, i.e., how it is represented, what is provenance for each result of every query executed.

8. ***Applicability of Framework*:** Applicability of proposed framework in terrorist attack investigation by identifying suspicious persons and their linked communities is also presented.

1.8.2.3 Key-Value Pair (KVP) Database

We designed and implemented BSDP (**B**ig **S**ocial **D**ata **P**rovenance) framework on top of ZILKVD. Qualitative analysis of existing provenance solutions and our proposed BSDP framework based on an evaluation matrix including data modelling, provenance granularity level, type of queries supported for provenance generation, provenance visualization, and its applicability is shown in Table 1.4. Main contributions of this work are:

1. ***Modelling Real-Time Streaming Twitter Data*:** We propose an algorithm to fetch a huge volume of real life social data from Twitter's network through live streaming by using Twitter Streaming API's.

2. ***Key-Value Pair (KVP) data model*:** We design an efficient Key-Value Pair (KVP) data model based upon a query driven approach to correlate this large size data through relationships and dependencies, in appropriate formats so that it makes sense for further analysis.

3. ***Querying Current State*:** Proposed framework allows querying the current data version and captures the provenance for the same. The captured provenance is stored in two column families viz., "select_provenance" and "update_provenance" for further analysis.

4. ***Querying Historical Data*:** An algorithm is proposed to query historical data by

Provenance Model	Data Model Design	Provenance Granularity	Provenance Capture						Provenance Visualization		Application Domain
			Select Query	Aggregate Query	Historical Query	Update Query	Insert Query	Delete Query	Justifying Query Results	Historical Data	
RAMP [100]	No	Workflow Level	Yes	Yes	No	No	No	No	Yes	No	Generic
Hadoop-Prov [106]	No	Workflow Level	Yes	Yes	No	No	No	No	Yes	No	Generic
Millieu [109]	No	Workflow Level	Yes	Yes	No	No	No	No	No	No	Generic
Layer Based Architecture [120]	No	Workflow Level	Yes	No	No	Yes	No	No	No	Yes	Generic
KVPM [116]	No	Fine-Grained Level	No	No	No	Yes	No	No	No	Yes	Application Specific (Patient Information System)
Our Proposed Framework (Big Social Data Provenance (BSDP))	Yes	Fine-Grained Level	Yes	Yes	Yes	Yes	Yes	Yes	Yes	Yes	Generic

Table 1.4: Qualitative analysis of different provenance solutions for Key-Value Pair Database with proposed BSDP Framework

extending Cassandra Query Language (CQL) with following constructs viz., "instance", "all", "validon now", and "validon date".

5. ***Provenance Generation in ZILKVD*:** Algorithms are proposed to generate provenance information for all queries including select, aggregate, historical, and data update queries with insert, delete and update operations, and to store captured provenance in ZILKVD.

6. ***Provenance Visualization*:** Framework also supports provenance visualization via querying provenance information for audit purpose, tracking all updates, and any other suitable application.

1.8.3 Provenance Query Templates

We propose the various provenance query templates to design an efficient provenance query language suitable for all types of databases without knowing anything about their internal data structures and implementation techniques.

1.9 Organization of Thesis

Chapter 2: Design and Development of Zero-Information Loss Databases This chapter explains the zero-information loss database concept with suitable examples. It also briefly describes about implementation of zero-information loss database in different databases i.e ZILRDB in relational database, ZILGDB in graph database, ZILKVD in key-value pair database.

Chapter 3: Provenance Framework for Relational Database This chapter presents the proposed DPHQ (Data Provenance for Historical Queries) framework on top of Zero-Information Loss Relational Database (ZILRDB). It provides the details about how ZILRDB is implemented, PRA (Provenance Relational Algebra) for provenance capture of relational queries, provenance storage in relational as well as graph database. It also provides details of Multi-Layer provenance capture and Multi-Depth provenance query support with suitable example queries.

Chapter 4: Provenance Framework for Graph Database Social Data Provenance (SDP) Framework on top of Zero-Information Loss Graph Database (ZILGDB) is presented in this chapter. It provides insight of modelling twitter data from csv file to Neo4j Graph Database, provenance capturing for query results including select, aggregate, data update queries, and querying provenance. Applicability of proposed framework in terrorist attack investigation by identifying suspicious persons and their linked communities is also presented in this chapter.

Chapter 5: Provenance Framework for Key-Value Pair (KVP) Database In this chapter, we propose an algorithm for modelling live streaming real-life Twitter data related to a specific event using Twitter Streaming API's efficiently in Apache Cassandra Key-Value Pair (KVP) Database. Big Social Data Provenance (BSDP) framework on top of Zero-Information Loss Key-Value Pair Database (ZILKVD) is presented in this chapter. For this, algorithms are proposed to design ZILKVD, and for capturing and storing provenance for select queries, aggregate queries, and historical queries. Provenance visualization support in proposed provenance framework via querying provenance information is also presented. An algorithm is also proposed for querying historical data efficiently.

Chapter 6: Template Based Provenance Querying In this chapter, we propose the provenance query templates based on all possible provenance queries in different databases, those may be used for implementing a provenance query language in future that can support almost all type of provenance queries efficiently in application and database independent way.

Chapter 7: Conclusion and Future Work In this chapter we conclude the thesis work and also present future directions for further work.