Abstract: Everyday, we are immersed in a vast amount of information without fully understanding the significance of it. Data generated from the ever-growing social networks, Internet-of-things and industries are extremely valuable resources in the Big Data Era, which practitioners are desperately trying to mine and to study. Studying and analysing such large amounts of data requires storing data from different sources with intrinsic relationship but having different formats and schema.

Despite of the dominance in market, relational databases are fairly limited when dealing such data, due to the restraints in the design of structure and schema. Graph Databases are increasingly gaining attention as a promising solution for linking data effectively. In this report, we will introduce Graph Databases and compare the characteristics of several mainstream Graph DB applications.

**Introduction:**

For many years now, relational databases have been in the forefront of the industry when it comes to storing data in a structured form, so that it can be queried, analysed, updated and maintained easily. Relational databases adopted the ideology of representing data in smaller tables which represent related content about a particular object. The tables are linked to one another using foreign keys. If the complete information of an object is required, we need to perform a join on all the smaller tables on the key attributes. This was believed to be an efficient approach, but in the current day with large volumes of data, it becomes increasingly difficult to perform these operations. Performing join operations (of the 6th or 7th order), on millions of records of the database is computationally expensive. Also, special care needs to be taken to enforce integrity constraints after every transaction. If a key value is changed in the parent table but not in the table referencing it, it can have fatal consequences. Above all, the complex structure of the schema design, table creation and query syntaxes add immense burden on the database programmer and make his job more difficult.

In recent years, newer technologies have emerged as an alternative to SQL. These so called ‘NoSQL’ technologies accomplish SQL and a whole lot more. There are 4 broad categories of NoSQL databases, based on the format in which the data is represented. The first category is Key-value pairs, and are similar in concept to a hash map. Redis is one such database. Column Families are another category, wherein the data is stored as columns instead of rows. Using this approach, the data is already indexed, and is very convenient for analytical processing. Eg- Cassandra. The third category is Document Databases, which view each row as a document, and store it in semi-structured formats like JSON. Eg- MongoDB. The fourth category is the Graph database, which represents data as graphs, i.e, a collection of nodes and edges. This representation is closest to real world entities, and convenient for modelling large data and studying relationships between objects. Our paper explores Graph databases in more detail.

**Overview of the Graph data structure:**

A graph is an abstract representation of objects & relationships. Each object can be viewed as a node, and a pair of nodes may be connected by links or edges. These links indicate how the objects are related to one another. Graph structures can effectively describe a lot of real world scenarios. Eg – a social network (links indicate who all are a particular users friends). Edges can be directed or undirected. (Eg – in twitter, A might follow B, but B needn’t follow A. Thus direction has significance). Graphs edges might also have weights assigned to them (eg – travel distance from city A to city B). Nodes and edges may also be labelled, or have property fields associated with them. Figure 2 shows a labelled property graph. There are 2 types of entities in the figure – people and books. There are 2 types of relations – write and purchase. The person node can have an additional label ‘author’ (since authors are also people). Among other things, the graph helps us infer that both Ian and Alan like the book titled ‘Tinker Tailor Soldier Spy’.

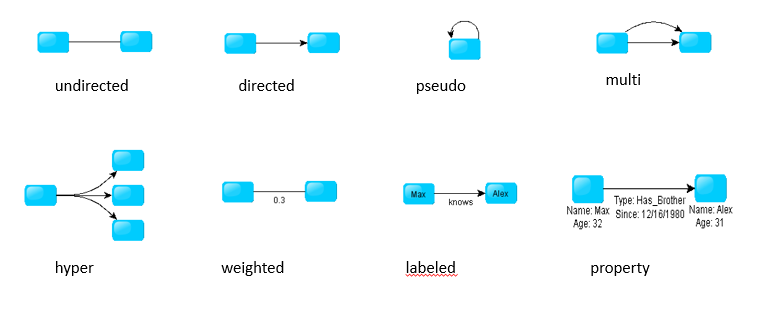


Fig 1 – types of graphs

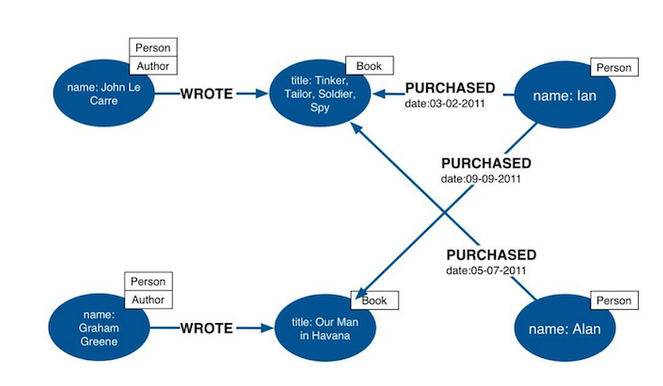


Fig 2 – labelled property graph.

**Graph Databases:**

A graph database is a database which has an explicit graph-like structure. An important point to note is that every node (record) is linked to its adjacent nodes directly, hence no indexes are necessary. As mentioned earlier, they are more suited to represent real world occurrences. Since there are no joins, it provides significant improvement in performance for large data sets. Also, the retrieval and traversal time is uniform irrespective of the breadth and depth of the search tree. Lastly, there is no rigid schema in the graph databases. Hence they can be modified or improvised easily, hence offer more flexibility and are convenient for ad-hoc projects.

There are a few key features that need to be offered by any graph database. Firstly, like any database, it should allow for storage of data in memory & disk. It should provide indexing where necessary. It should allow for representation of data in multiple formats. It should support DDL (Data Definition Language, i.e creation of schemas, tables), DML (Data Modification Language, i.e insert, delete, update records) and DQL (Data Query Language). The queries in graph databases are vastly different from traditional databases. It should allow querying for adjacency (whether 2 nodes are directly connected or not), reachability (does a path exist between 2 nodes?) and shortest path computation (finding the optimized route when multiple alternatives exist). It should also allow for aggregation and summarization. A good add-on will be APIs to plug in components from other languages.

The most popular Graph database currently in existence is Neo4j. It is an open source tool, and runs over the Java virtual machine. It provides indexing using Lucene. Data is represented as property graphs. It is easily scalable for large data. It supports REST APIs, and can be used with other languages like R or python. It provides inbuilt query languages like Cypher & Gremlin, for pattern matching and traversal.

Some use cases where neo4J can be widely applied include – social networks and other widely connected graphs; product recommendations and dating apps (Find movies that X likes, then find others who also like those movies. Find other movies that they liked and recommend them to X. Or, find out all women who share the same interests as X and live within a 10 km radius and show them to X); Finding if path exists or not (if X wants a job at company Y, find if he has connections who can refer him to other people at company Y, so that he is closer to getting the job); Finding the shortest path and optimizing the travelling salesman problem.

Conclusion:

This report listed the key drawbacks of relational databases, and explored graph databases as an alternative. Given their usefulness and convenience for large scale applications, Graph databases should be a vital tool for data management, querying an analysis in the near future. However, this is only an emerging field and most commercial databases only offer a subset of the desired functionality. They are still a ‘work in progress’ , and there is much scope for improvement. Looking at the benefits which are on offer from the use of graph databases, more and more people might adopt them in the coming years. It may not be a full time replacement for SQL and relational databases, but can surely form a larger piece of the pie.