IN3020

Assignment 2, 2021

Kim-Reino Hjelde

Kimreinh

Exercise 1:

a)

T1 = $r_1(A);r_1(B);w_1(A);w_1(B)$

T2 = $r_2(A);r_2(B);w_2(A);w_2(B)$

b)

Correct: $r_1(A);r_1(B);w_1(A);w_1(B);r_2(A);r_2(B);w_2(A);w_2(B)$

Incorrect: $r_1(A);r_1(B);r_2(A);r_2(B);w_2(A);w_2(B);w_1(A);w_1(B)$

The correctness criteria I have in mind for concurrent transaction schedules is that they are equivalent in the outcome of database states to serial transaction schedules, meaning that they are equivalent to a transaction schedule that is fully sequential. This is strict and demanding, but goes a long way to ensure atomicity, consistency, isolation and durability.

The second example is incorrect because it violates consistency with a read-write conflict, an unrepeatable read anomaly. Unlike the first, it is not serial or serializable. T1 and T2 both read original values, then T2 overwrites and commits, which leaves T1 with an inconsistent understanding of the state of A and B.

c)

S_A: $r_1(A);r_1(B);w_1(A);w_1(B);r_2(A);r_2(B);w_2(A);w_2(B)$

S_B: $r_1(A);r_1(B);r_2(A);w_1(A);w_1(B);r_2(B);w_2(A);w_2(B)$

S_C: $r_1(A);r_1(B);w_1(A);r_2(A);w_1(B);r_2(B);w_2(A);w_2(B)$

d)

S_A is already fully serial, as each transaction is listed sequentially as T1;T2.

S_B is neither serial nor serializable because of the subsequence**(**$r_2(A);w_1(A);$**)** which cannot be resolved through exchanges of neighbour operations without conflict.

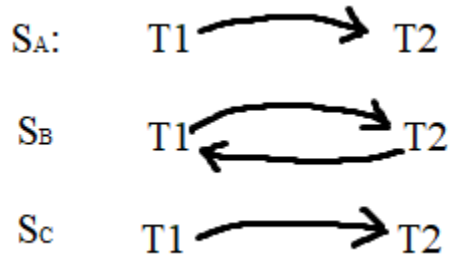
S_C is serializable because it is conflict equivalent to S_A, which is a serial execution plan. This is achieved by switching the operations $r_2(A);w_1(B)$.

1.2

a)

There are read-write conflicts, write-write conflicts, and intra-transactional conflicts. Read-write conflicts and write-write conflicts are rather similar in that they share conditions, namely that a pair of operations is in conflict if they are from different transactions AND they operate on the same data element AND at least one write operation is involved. Intra-transactional conflicts on the other hand have to do with resolving plans into serial execution plans, where this resolving through neighbor exchanges is impossible because it would violate the order of the operations within the relevant transaction.

b)



c)

When determining which execution plans are conflict serializable from looking at precedence graphs, one should look for cycles. The presence of a cycle implies that it is not conflict serializable, and as such, only acyclic graphs are accepted.

d)

2PL is basically a good compromise between maintaining ACID properties and allowing for concurrency. With an expanded set of rules compared to simple locks, one can ensure conflict serializability with the growing-phase --> shrinking-phase setup. The downside for concurrency is that a transaction may not be able to release an item quickly after having used it. 2PL also does not prevent deadlocks.

Exercise 2:

2.1)

In multiversion databases, multiple versions of data elements are stored and created rather than being overwritten. This requires that transactions get timestamp identifiers, and a mechanism for dealing with old and unused garbage values. Snapshot Isolation is one such protocol for managing multiversion databases. At the start of a transaction, it will be given a kind of personal snapshot of the database to work with. In practice this means that each transaction T is only working with data written by transactions before T started. Further, the transaction T will only successfully commit if the data that is being updated by T has not been modified by a transaction external to the current transaction T, that is, outside the snapshot of T. Thus, the write-set of a number of concurrent transactions can not overlap.

SI is popular because it is a way of implementing a robust multiversion-concurrency-control system, and MVCC is a common way to increase concurrency(and thus performance) while avoiding many concurrency anomalies that serializability also avoids, but not all. The three anomalies that may occur with SI protocol are phantom skewed writing, skewed writing, and read transaction anomalies. Since skewed writing anomalies can occur in SI, it does not guarantee serializability. Serializability is a strict standard, however, and since SI avoids the majority of anomalies that occur in less strict isolation levels by SQL-92 standards, it remains a very viable alternative.

2.2)

An SI plan is any merging of a set of transaction plans in accordance with the SI protocol. By strict, it is meant that any SI plan adheres to the part of the SI protocol that demands that the write-set of a set of concurrent transactions is disjoint. This means that any transaction T that writes a data element X must successfully commit or abort before any other transaction may access X.

2.3)

Postgres:

- MVCC, serializable snapshot isolation, read never blocks write and write never blocks read due to MVCC, PSQL implements three distinct isolation levels instead of the four from the standard due to how the levels map onto the MVCC-architecture.

Exercise 3:

a)

The key driver in the development and expansion of NoSQL databases is the need for DBMSs that can better support data that is stored and processed on a much wider and unstructured scale than the rather vertical setup that SQL historically was built to handle. The kind of information that data science, AI/ML, and Big Data applications in general handle, is impossible to organize into orderly tables due to quantity and formats, and so NoSQL databases are designed to handle data in the form of documenst(JSON, XML), key-value pairings, graphs etc. to meet this need.

b)

A schema in DBMSs such as MySQL is basically a fixed data structure that the tables in the DBMS must adhere to. This means that the schema defines what type of data can be stored, as well as where. In a NoSQL database on the other hand, the responsibility of deciding the what and where of storing data is left to the developers. That a NoSQL database is schemaless, then, means that there is no pre-specified data structure such as a table with x columns and y rows that take dates or formats or strings, but rather a dynamic collection of 'raw' information, such as MongoDB with JSON documents that do not necessarily follow the same structure.

c)

Document-based: A document means a self-describing collection of objects that have varying attributes and data elements but are overall similar, commonly presented as XML or JSON objectives.

Key-value pairs: A system based on managing data in a structure like a hash-table, with dictionaries containing a collection of objects/records which in turn has different fields.

Column-oriented: Multidimensional keys leading to named tables with self-describing rows. Related columns are grouped together in column families which are created with the table itself. A given column family is associated with a column qualifier. Good for storing large amounts of data, great data aggregation and search.

Property graph(labelled): Labelled nodes have inner structure that describes properties, and edges give relationships between nodes. Edges may also carry structured information. Labels denote the type of the node, and each node can have several labels. Properties in nodes and relationship edges are key-value

pairs. Graph traversal done with pointers, which is much cheaper than joins. A graph database is dynamic, and workload is shifted from query execution to general data maintenance.

RDF-based graph: Based on the semantic RDF framework, with triplets on the form of <subject, predicate, object>, stored as RDF documents with special syntax. As a graph it is represented as a directed labeled graph, where nodes are subjects and objects, and edges are predicates. RDF allows for making semantic statements about the content of a triplet, known as the resources in a triplet. Resources and relationships that are not literals are identified by URIs which are unique labels, and they offer no internal structure.

Multi-model: a database that contains multiple data storage mechanisms from the ones listed above in the same DBMS architecture.

d)

They are both examples of NoSQL databases based on the core of column-oriented DBMS architecture with their own proprietary expansion on the core concept. In Hbase, a database only has a schema in so far as the tables only define column families on creation. The tables are collections of sorted rows, and each row has multiple column families defined at creation. Column families are flexible in the sense that additional columns can be added to a family at any time. Each column has two components, namely the column family and a column qualifier. Further, each cell value has a timestamp. The architecture can in this sense be seen as key-value pairs, where one can specify RowKey:ColumnFamily:ColumnQualifier:Version.

e)

Advantages: Runs and scales well with low-cost commodity hardware, flexibility of dynamic schemas/schemaless, solutions for a wide range of data formats and types.
Disadvantages: Little to no support for the traditional ACID principles, not compatible with the already established SQL infrastructure, lack of maturity and support(requires more expertise to successfully run and maintain)

3.2

a)

The CAP theorem states that a tradeoff must be made between providing consistency, availability and partition tolerance in a distributed system. This is very relevant for NoSQL databases which are geared towards distributed systems, seeing as the tradeoff mostly comes into play when partition tolerance is relevant, and so usually one of the three goals must be sacrificed. By partition tolerance, it is meant that a system must continue to work despite a communication breakdown between two nodes in the system. When a communication fault occurs, the inconsistent node must be made unavailable until it is made consistent again, in which case availability is compromised. On the other hand, if the inconsistent node is kept available, queries might return an older version of data compared to other nodes, which means a resync will be necessary down the line for repair inconsistencies. For a distributed NoSQL system to promise both availability and consistency it must never have any issues with communication between any nodes in the system, because that would create a partition, and this is an unrealistic expectation.

b)

It is not possible for a NoSQL database of any meaningful scale to be fully ACID-compliant due to the CAP theorem and I have outlined my understanding in 3.2 a.

c)

I included RDF and property graph databases briefly in 3.1 c. I will supplement those summaries here. One difference between RDF and property graphs, are the nature of relationships between nodes. Relationships between nodes in property graphs have their own structure and attributes, and can be uniquely identified, whereas in RDF there are no unique instances of relationships of the same type, and so an intermediate node is created to facilitate representing the additional information. This greatly complicates the query as well as the visual representation of the graph. In a property graph, a node of the type Person could contain basic information about that person without needing additional auxiliary nodes to facilitate this. Property graph databases lack standardization and have many implementations, the most known of which is Neo4J with its own query language cypher. RDF on the other hand is part of a stack managed by W3C, which means it has a well established standard on the web, with the query language SPARQL.

d)

Data Quality Management is a system that aims to ensure that data used by applications is of high enough quality in regards to its purpose. There are many aspects to take into consideration when evaluating

whether data is of “high quality”, and the standards for quality can differ between various applications and use cases. Generally, data should be of the right type and format, and key fields should not be empty. Further, data quality can be measured by its accuracy, completeness, reliability(matching trusted sources?), relevance(superfluous?) and timeliness(up to date?).

e)

ETL and ELT are acronyms for extract-transform-load. It is essentially a data integration process wherein data is sourced(E) from multiple locations and copied(L) into a destination system for some purpose(T). That may be to create a single consistent data store, or it can be to present the data differently from the sources, for example. The difference between the two is whether the data is transformed(cleaned, formatted etc.) before or after it is loaded.

For the stated purpose, both ETL and ELT sound like viable ways to build a big collection of data. The main difference between the two would come down to whether I would need to clean and otherwise transform the data beforehand out of privacy concerns, or if I would end up preferring having access to a lot of raw unprocessed data to explore. I would Extract data from various sources, Transform it by cleaning and formatting into perhaps CSV format, then Load it into an existing python framework such as pandas and go from there.

Exercise 4:

4.1

a)

An important incentive for any entity handling data is compliance with legal frameworks such as the GDPR and national laws, and avoiding massive fines. In the context of ensuring data security and privacy in DBMSs, it is important to look at the full scope. There are clients, servers, physical components and personnel, and individually these are all potential vectors for attacks and breaches. Additionally, these all interact together, and protocols are needed to make sure that the security of the system as a whole is maintained. More specifically, the classic trifecta of Confidentiality, Integrity and Availability, must be ensured.

Robust authorization protocols ensure that entities that should not have access to data cannot view or modify it in any way, this is in the spirit of confidentiality and integrity. Need-to-know access, the concept of least possible privilege, avoiding privilege creep, having a role-based or attribute-based access control system.

b)

SQL injection: Wherein malicious SQL statements are inserted into entry fields to be executed. This is an exploit that targets the software of a database system, and takes advantage of sloppy 'sanitation' of user input.

DB traffic sniffing: packets sent between components or nodes in a system can be 'sniffed' and analyzed by malicious entities with access to communication channels.

Abusing privileged accounts: An entity with malicious intent can do damage if it has access to privileges that it shouldn't have, whether obtained by an exploit or through negligence, such as excessive granting of privileges or failure to take privilege creep into consideration.

c)

In addition to the good practices mentioned previously which amount to keeping track of accounts with their respective privileges and having an access control system which takes into account the security level and class of various data, the use of encryption for both communication and data is important, and should be baked into the system.

Two ways of providing access control which most DBMSs offer are label based security on the level of individual rows, which is rather fine grained compared to having labels for entire relations, and providing access control on an account level, which in comparison is rather coarse.

d)

In identifying an individual, there are a few singular attributes that must be anonymized. Examples of this would be full name, or national identification number such as a birth number. It would be a simple matter to transform cells containing this information into empty dummy values before sharing. There are also personal attributes that, when considered together, can become personally identifiable. In an academic setting, this could be the combination of date of birth, first or last name, and classes that the person participated in. It seems to me to be a very context-dependent thing to determine whether there is a breach of privacy or information confidentiality. Information security on the other hand would be a matter of hosting the data on the internet responsibly, ensuring that the data is not modified after distribution for example. Further, if there is a specific set of researchers who are supposed to have access, it would be a

security breach rather than a privacy breach, if someone unauthorized were to access it and assuming that the data has been sufficiently pseudonymized.

4.2

a)

Proper role and identity management is key to avoiding unauthorized access of data. It entails having an authority(an administrator equipped with relevant protocols, information and tools) in charge of defining roles and identities given to accounts, as well as what privileges each role has. Privileges, and thus access, must be granted sparingly and revoked as soon as the purpose for granting it in the first place is no longer relevant.

b)

Oracle-based security architecture uses a fine grained labelling system for users and data down to the level of rows, and is strictly hierarchical and enforces multiple levels of sensitivity and security for data. Each row in a table may have multiple labels, which together denote the minimum access level required by users to access the data. Individual users each have labels which denote the maximum access level the user has access to, and so a very specific set of conditions must be met before real access is granted to any row in question. Labels range from unclassified to highly sensitive.