IN3020 Assignment 2

Espen Lønes

May 9, 2021

Exercise 1)

1.1)

a)

T1
Read(A)
A ← unavailable
Write(A)
Read(B)
B ← unavailable
Write(B)

T2
Read(A)
A ← unavailable
Write(A)
Read(B)
B ← unavailable
Write(B)

b)

T1	T2
Read(A)	
	Read(A)
A ← unavailable	
	A ← unavailable
Write(A)	
	Write(A)
Read(B)	
B ← unavailable	
Write(B)	
	Read(B)
	B ← unavailable
	Write(B)

T1	T2
Read(A)	
A ← unavailable	
Write(A)	
	Read(A)
	A ← unavailable
	Write(A)
Read(B)	
B ← unavailable	
Write(B)	
	Read(B)
	B ← unavailable
	Write(B)

In the first (left) plan T1 and T2 are able to read an mark unavailable / reserve the same item. While with the second plan T1 has already marked its A before T2 attempts to read.

c)

T1	T2
Read(A)	
A ← unavailable	
Write(A)	
	Read(B)
	B ← unavailable
	Write(B)
Read(B)	
B ← unavailable	
Write(B)	
	Read(A)
	A ← unavailable
	Write(A)

T1	T2
Read(A)	
	Read(B)
A ← unavailable	
	B ← unavailable
Write(A)	
	Write(B)
Read(B)	
	Read(A)
B ← unavailable	
	A ← unavailable
Write(B)	
	Write(A)

T1	T2
Read(A)	
Read(B)	
A ← unavailable	
B ← unavailable	
Write(A)	
Write(B)	
	Read(A)
	Read(B)
	A ← unavailable
	B ← unavailable
	Write(A)
	Write(B)

d)

The last one is serializable. The first has write-write and read-write conflict. The second has read-write conflict. The third has no conflict.

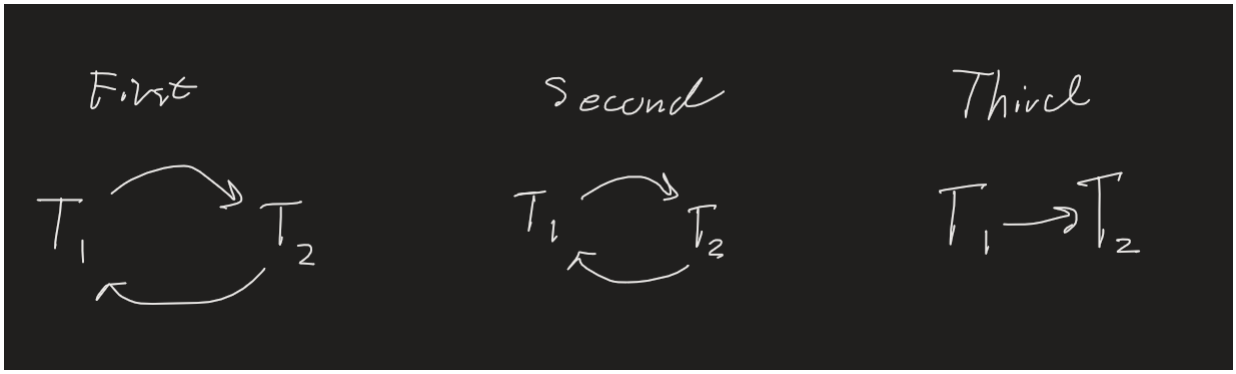
1.2)
a)

write-write:
Two transactions writing to the same element.
$...W_i(A)...W_k(A)...$, $k \neq i$

read-write:
One transaction reading and one writing on the same element.
$...W_i(A)...R_k(A)...$ or $...R_i(A)...W_k(A)...$, $l \neq i$

intra-transaction:
An operation on two data elements within the same transaction.
$...O_i(A)...O_i(B)...$, O_i is W_i or R_i

b)



c)

The ones that are cyclic are not conflict serializable. But the Third one has no cycles and is conflict serializable.

d)

2PL helps us ensure conflict serializability. An item can only have one lock at a time meaning only one transaction at a time can change it.
This and the other two rules of 2PL ensures a plan is conflict serializable ($\iff$ serializable) meaning, by enforcing the rules of 2PL there is no need to check for serializability as they always will be.

Exercise 2)
2.1)

Snapshot isolation creates multiversion plans. Snapshot isolation has two rules.
1. A transaction reads the latest version of an item that was committed before the transaction itself started.
2. If one transaction starts before another and ends after the other has started (They do some operations simultaneously), they can not write on the same item.

Conforming to snapshot isolation does not mean that a plan is serializable. But it is still quite strict, only allowing for these three anomalies:
A3B - Phantom skew writing
A5B - Skew writing
A6 - Read transaction anomaly

Making it Stricter than read committed, not as strict as Seizable. And neither stricter or weaker than repeatable read.
SI is used when a DB does not have a need all plans to be 100% serializable. Giving the planer room to make more interleaved/parametrized plans making it possible for greater DB speed.

2.2)

An SI plan in a plan that follows the two rules mentioned in the previous task. All SI plans are strict meaning if a transaction writes to a data item, the transaction must either commit or abort before other transactions can read or write on the same item.

2.3)

Postgres:
Has the following table explaining what is allowed in the different isolation levels:

Isolation Level	Dirty Read	Nonrepeatable Read	Phantom Read	Serialization Anomaly
Read uncommitted	Allowed, but not in PG	Possible	Possible	Possible
Read committed	Not possible	Possible	Possible	Possible
Repeatable read	Not possible	Not possible	Allowed, but not in PG	Possible
Serializable	Not possible	Not possible	Not possible	Not possible

In Postgres all four of the standard isolation levels can be requested but only three are implemented as their read uncommitted behaves like read committed. We can also see that their repeatable read does not allow for phantom read. Read committed is the default.

MySQL:
Provides all four of the standard isolation levels (by the SQL:1992 standard). Read committed, Read uncommitted, Repeatable read and Serializable. The default option is Repeatable read.

Oracle:
Has the following table explaining what is allowed in the different isolation levels:

Isolation Level	Dirty Read	Unrepeatable Read	Phantom Read
READ UNCOMMITTED	Possible	Possible	Possible
READ COMMITTED	Not possible	Possible	Possible
REPEATABLE READ	Not possible	Not possible	Possible
SERIALIZABLE	Not possible	Not possible	Not possible

Oracle provides Serializable, Repeatable read and Read committed (default). But does not support Read uncommitted.

We see that Postgres and Oracle in practice provide the three same isolation levels and have the same default level. Unlike MySQL that provides all for as explained by the standard and has repeatable read as default.

Exercise 3)
3.1)
a)

Many systems that e.g. stores large quantities of data need DBMSs that fit their needs. The traditional SQL DB does not do this. Most of the NoSQL DB systems are distributed DBs with focus on semistructured data, scalability, eventual consistency, data replication, high performance and availability.

b)

A DB schema is its structure described in a formal language. The schema is the organization of the data as a blueprint of what the DB 'looks' like. e.g tables for relational DBs. The formal definition is a set of integrity rules.
A DB schema is usually made/defined on creation of the DB and not expected to change much. NoSQL has no or partial schemas. This requires more work when setting constraints but opens for a lot more flexibility of the DB structure.

c)

Document-based:
Stores and retrieves data in the form of documents. e.g JSON or other standardised formats. Documents are indexed via document id, but can also use other indices. Relies on internal structure in the document to extract metadata used to optimize search.

Key-value pair / Key-object pair:
Stores and retrieves associative arrays aka. dictionary and hash table data structures. Contains a collection of objects or records which themselves have many different fields within them, where these fields contain the data. The records are retreved using a key that has a unique id. These DBs work in almost real time and are good for capturing time-series data.

Column-based:
Stores data tables by column partitioned into column families. each column family is stored in its own files. These DBs also allow data versioning and are good for storing large amounts of data.

Graph-based:
Uses graph structures to execute semantic queries, with nodes, edges and properties to describe the data. Allows data to be liked directly and in many instances retrieved with one operation. Relationships can be easily visualised, meaning they are useful for heavily inter-connected data. They can be used for pattern matching.

d)

They are column based, and used for storing large amounts of data. Goggles Bigtable uses Goggle FIle System. Apaches Hbase is simelar. They have multidimensional keys. They are often called sparse multidimensional distributed persistent store map.

Hbase:
Stores semi-structured data with different data types, varying column and field size. The layout of the HBase data model eases data partitioning and distribution across the cluster. The data model consists of many logical components, row key, column family, table name, timestamp, etc. Row keys are used to identify unique rows in tables. The column families are static whereas the columns are dynamic.
The HBase tables are logical collections of rows stored in partitions called regions.
Rows are instances of data in a table.
All entries are identified by a row key.
For every row key an unlimited number of attributes may be stored. (columns)
Data in rows is grouped together as column families. All columns are stored in a low level storage file (a HFile).

e)

NoSQL DBs have much higher performance, a higher degree of availability and data replication. They also handle scalability better than traditional (SQL) DBs.

They lack the advantages of structured data storage and consistency. SQL DBs often have more powerful query languages.

3.2)
a)

CAP theorem states that it is impossible for a distributed data base to provide more than two of the flowing guarantees simultaneously:

Consistency (among replicas): Every read receives the most write or an error
Availability: Every request reeves a non-error response, without guarantee it has the most recent write.
Partition tolerance: The system continues despite any number of dropped or delayed by the intra-node network.

The CAP theorem says if we have a partitioned network we must chose between consistency and availability. (Not the same as ACID consistency).

b)

There exists distributed concuracy control methods that enforce serializability and disallow inconsistency between copies. But enforcing serializability is the strongest form of consistency and caries with it high amounts of overhead. Greatly slowing down read and write speeds. But the lack of ACID in a NoSQL DB to have greater speed is kind of the whole point of a NoSQL DB. So yes we probably can make NoSQL DBs conform to ACID but then we should just consider using a SQL DB instead.

c)

RDF and Property graph databases are both graph DBs. RDF is older than Property graph, because of this it is standardised and also has a standard query language (SPARQL). Property graphs on the other hand are newer and not yet standardised (is going to be standardised in the future). It also has a number of different query languages (PGQL, Cypher). The main use of PG is to do pattern matching.

Property graphs are represented by a set of nodes, relations, properties and labels. Nodes, data and their relationships can have their own properties. RDFs are mush alike representing the parts as triples
< subject(node/data), predicate(relationship), object(node/data) >.
A significant difference is that a RDF can't uniquely identify relationship instances, i.o.w, connections of the same type between the same nodes will result in exactly the same tipple. Where as in Property graphs the relationships themselves can have properties.

In RDFs we can have multivalued properties, triples where subject and predicate are the same but the object has changed. To get the equivalent in a property graph we must use arrays.

RDFs have something called a quad. Which makes it possible to add context or extra values to a triple. This makes it easier to make subgraphs or named properties. Property graphs have no equivalent to this.

d)

Data quality management is used to ensure the quality of data omitted to the DB or data already in the DB. As any application relying on data, requires data of high (enough) quality, and may fail or provide incorrect information if using low quality data. DQM systems have five key features it should ensure.

Accuracy: Is the data correct.
Completeness: Is the data comprehensive enough.
Reliability: Does the data contradict with other trusted sources of data.
Relevance: Do we have use/need for this data.
Timeliness: Is the data new/up-to-date. Can it be used real time.

e)

ETL and ELT stand for respectively, Extract-Transform-Load and Extract-Load-Transform.

Extract: Pulling data from its source.
Transform: Changing the data to fit the requirements of the DB it is going to. Done by using buisnies rules, lookup tables or combining with other data.
Load: Writing to destination DB.

While working on this type of project. ELT or ETL can be used to gather, restructure and centralize data about Oslos housing market. To make it possible for the ML to use the data.

Exercise 4)
4.1)
a)

The point of DB security is that DBs can contain sensitive, personal or valuable data. That the owners and users of the DB don't want just anyone to have access to.

The main idea of DB security is to prevent unauthorised use or unauthorized access. Key components to ensure this are:

Need-to-know: Have access to the data you require to do your job and only that data.
Least privilege: You only have the minimum amount of privileges needed.
Privilege creep: As your role changes, new privileges are granted. Privileges no longer required aren't revoked (as they should be).
Governance and access control: Part of the system responsible for granting and revoking privilege/access.
Role based access control: Logical group containing a group of access rights. Thereby granting roles instead of individual access rights.
Attribute based access control: Looks at dynamic factors like time of day and location. To decide if rights or roles are granted/revoked.

b)

Integrity loss: Unauthorized changing of data. Like creating, deleting, modifying/updating data. Like changing the amount in a bank account or zeroing out a loan. Can be done with SQL-injections

Loss of availability: Authorized users cannot access/use the data. e.g. after a DDoS attack.

Loss of confidentiality: Unauthorized/unintended disclosure of data. Like leaking personal files.

Other types of security breach methods are ransomware, databreach and sniffing.

c)

Encryption, symmetric key algorithms, public keys, digital signatures, digital certificates, firewall.

d)

We can in some way anonymize the data, either by removing or altering the parts of the data that is used to identify, in a way that doesn't hider its usefulness for the researchers. We could also use the real data to generate dummy data resembling the real data but isn't actually the real data.

4.2)
a)

Identity management is a framework of policies used to ensure that the correct users have appropriate access to technology resources. This controls information about users computers, like information about the identity of a user, what data and actions they are authorized to access and/or preform. It also includes the management of the user and how and by whom that information can be accessed and altered. A role is a predefined collection of allowed actions and data access, that can be granted users. And that is required for a user to access data and or do any of the related actions.

b)

The oracle based security architecture is unique to oracle systems. It can restrict access from table level privileges down to row specific access control. Providing very fine control of user access rights on a database. It uses user labels describing the maximum access level a user has. And compares these Data labels, which are the minimum access level needed to have authorized access to the data. Like expressed in this illustration.