Remy Lagrois

DB Final Section 402

1. **e) None of the above.** RDMBS can only handle simple datatypes like integers and strings. Storage of the types of data listed in this question are part of the reason for the emergence of NoSQL (Asynch unit 13 and book chapter 7)

2. **d) All of the above.** NoSQL is simpler than a traditional database as no schema is needed so there is less planning needed in the design phase. Relational databases need data that conforms to its table format whereas NoSQL can take many types of data as is. NoSQL is also flexible since as mentioned it is schema-less and there are also many types of NoSQL databases (document, graph, key-value, etc). (Asynch unit 13 and book ch 3)

3. **b) Key/Value pairs, Key/Value pairs.** Both the input and the output are key/value pairs according to the paper (MapReduce paper)

4. **c) Each instance of the reduce function receives all of the intermediate data from every map function.** The paper explicitly says this happens. The other two are wrong though because the data is broken up into chunks before being sent to the map workers. The map and reduce functions are performed in an specific ordered way by dedicated workers and are not chosen randomly (Map Reduce Paper)

5. **a) The map intermediate results are stored on local machines.** According to the paper the map intermediate results are in fact stored locally. A combiner function can be used on the map worker machines which manipulates the data before it is passed to the reduce workers. The results of the reduce function are sent back to the user. (Map Reduce paper)

6. **d) All of the above.** Details of parallelization are hidden which makes it easy to use for programmers with little experience. It is also efficient and simple due to the way the data is broken up and tasks are assigned. It also can be used for many problems such as sorting, machine learning, and data mining. (MapReduce paper)

7. **e) All of the above.** BigTable is highly distributed over many computers which makes it scalable (add more computers), available (resilient to total system failures), and gives it high performance (distribute traffic and loads). It is also widely applicable as can be seen in the number of projects Google has used BigTable. In fact, all of these attributes are listed in the introduction as the goals for big table. (BigTable Paper)

8. **c) BigTable provides a simple data model that supports dynamic control over data layout and format.** BigTable is designed to be flexible and applicable to many projects. This is accomplished by a simple data model that the user has a lot of control over. (BigTable Paper)

9. **d) All of the above.** In BigTable both reads and writes are atomic for a single row key. Access control rights are also available at the column-family level (BigTable paper).

10. **b) 3** Level 1 is the root tablet which contains the location of the metadata tablets. Level 2 is the metadata tablet and contains the location of the user tablet. Level 3 is the user tablet itself. (BigTable Paper)

11. **b) Multiple authors.** Book title and publication date are exactly what would be expected in such a table. Having a multi-paragraph attribute value is not ideal since it would make each row very large making it difficult to view many records at once. It also would probably be difficult to select based on the abstract attribute. Having multiple authors in a single attribute would be removed during normalization so most of the time you would not see an attribute with multiple values. (Book ch 8)

12. **d) All of the above.** The actual structure of MongoDB documents make all of these viable. MongoDB also uses BSON which allows all those types of data to be stored. (Asynch unit 14)

13. **a) The normal case.** Amazon runs so many machines in so many different locations that things are constantly going wrong just due to the law of averages. Thus they must be treated as normal and worked around.(Dynamo paper)

14. **b) Primary key** Most operations (like the shopping cart, best seller list, product catalog) only require the primary key. More complex functions take more hardware and expertise. Limiting dynamo to primary key access meets the needs, is cheaper, and is more efficient. (Dynamo Paper)

15. **d) Consistency.** Consistency on Dynamo is eventually consistent. Conflict resolution is done at the read stage so writes are never rejected. This means that if a customer adds something to their cart it won’t ever disappear when they go to check their cart (Dynamo Paper)

16. **a) Uniform data distribution through consistent hashing**. The nodes form a “ring” which means when one node goes down it only affects the adjacent nodes. Amazon takes this further by using what they call a “virtual node” where one node is responsible for multiple parts of the ring. This means that the load is evenly distributed; when a node goes down the other nodes can take an equal amount of the load and when a new node becomes available the load it takes is about equal to the other nodes. (Dynamo Paper)

17. **d) It is impossible for a distributed database to achieve consistency, availability and partition tolerance simultaneously.** According to the CAP theorem you can have two of these simultaneously but not all three. In order to achieve two the third must be sacrificed. For example you could have high availability and partition tolerance but that would be consistency would have to be much less strict. (Asynch Unit 13)

18. **b) they are designed to reap the read and write performance benefits of partition tolerance (horizontal scaling) while** Horizontal scaling is one of the main goals for using NoSQL databases, almost all of them allow for good partition tolerance. The NoSQL databases systems are more split when it comes to choosing the second strong attribute, it depends on the function of the database. Some, like MongoDB, are consistent and partition tolerant, while others such as Dynamo are available and partition tolerance. Horizontal scaling allows for rapid growth much more cheaply since regular consumer machines just need to be plugged into the system. (Asynch unit 13 and <http://blog.nahurst.com/visual-guide-to-nosql-systems>)

19. **d) All of the above.** MongoDB is not ACID compliant and is only atomic at the individual document level. It does however have an expressive language which is summed up by the CRUD acronym (create, read, update, and delete). Despite not being ACID compliant it does have consistent reads and as mentioned before is atomic at the document level. (Asynch 14, <https://docs.mongodb.org/manual/faq/fundamentals/>, <https://dzone.com/articles/how-acid-mongodb>)

20. **d) All of the above.** As mentioned in question 19 MongoDB does ensure atomicity for a single document but not for operations that span across multiple documents. Inserting, updating, and removing data will put a write lock on the document to ensure atomicity and consistency. That does mean though that if two people are trying to write different things to the same document at the same time one of them won’t be able to. (<https://docs.mongodb.org/manual/faq/concurrency/>)

21. **a) Relational type.** Relational databases would work best for this application since the data is very predictable and well structured. Relational databases are also ACID compliant which is important for banking since money is on the line. Transfering from one account to another must be atomic and durable or else money could be taken out of the account but never placed in the other. If the server crashes as soon as the transfer happens and the database wasn’t durable then the money could be lost again. (<https://dzone.com/articles/how-acid-mongodb>)

22. **b) More writes than reads.** Cassandra is well known for its fast writes. A write is sent from a client to a random node which then sends it out to a number of other nodes. The write is stored in memory until the “memTable” is filled then it writes to the disk. Having multiple nodes take the write also increases redundancy. (http://www.mikeperham.com/2010/03/13/cassandra-internals-writing/)

23. **a) Sharding.** Distributing data across multiple machines is called sharding. This is how MongoDB grows, that is to say adds more storage. It also can help with availability, if one machine crashes then only the data stored on that particular machine becomes unavailable instead of the whole database. (<https://docs.mongodb.org/manual/sharding/>)

24. **d) Facebook.** Cassandra was initially developed at Facebook using Dynamo and BigTable for their inbox searching feature. Now though it is open source. (<http://www.planetcassandra.org/what-is-apache-cassandra/>, <https://en.wikipedia.org/wiki/Apache_Cassandra>)

25. Advantages:

1. Scalability: NoSQL databases are designed to be highly scalable to handle rapid growth in data storage needs by adding commodity machines.

2. High Availability: Because NoSQL databases are highly distributed they can handle high traffic and partial system failures.

3. Variable data types: SQL databases can be limited in the types of data they can handle while NoSQL databases are much more accepting of unknown or “blob-like” data.

4. Low Cost: Since NoSQL typically run on commodity machines they can be grown much more cheaply than SQL databases. They also are often open source so there are no licensing fees.

5. Customizable: There are many different types of NoSQL databases which can be used to fit many different needs. They are also often schema-less so can be changed as needed.

Disadvantages:

1. Not ACID compliant: RDBMS databases are ACID compliant which means operations performed on them are very secure and reliable compared to NoSQL.
2. Very New: RDBMS has been used for many years so there is a wealth of expertise and knowledge that doesn’t exist yet in NoSQL.
3. Queries can be difficult: Queries can be more difficult on NoSQL since it doesn’t have the same rigid table structure as RDBMS and also stores types of data that may not have an easy way to search for.
4. CAP: NoSQL can only ever be good at two out of three (availability, consistency, partition tolerance). Relational databases are vertically integrated so these decisions need not be made.
5. Difficult to index: This is related to the difficult queries due to the structure and variable data types.