CSCU9YQ: NoSQL Databases

2016 Exam Answers and Marking Scheme

Question 1

This question is mostly book work, but requires reasoned answers rather than repeated facts.

a) A schema imposes a structure on the data being stored. It defines the tables in which data are stored and, for each table, the names and data types of the fields it contains (among other things). Many NoSQL databases do not enforce a schema, meaning that each entry (in a collection, taking MongoDB as the example) can have different fields from the others. Fields are defined at run time, rather than design time (though a design can be imposed at the application layer). 4 marks for a description like that, then two for an example such as a collection of 'people' where all documents store a name, but some store facebook page, others linked in page, some both, etc.

b) –
   i. Replication copies data into more than one place. Its main purpose is resilience. Sharding spreads the data across a cluster, with a portion of the DB on each node. Its purpose is speed and scalability.
   ii. A shard key determines which node a document is sent to (1 mark). It should be a field that appears in every document and be divisible into a number of values that is at least as many as the number of nodes in a cluster. It can have a null value, and can be compound. 2 marks for two such points.
   iii. Numeric fields are sharded according to a range (1) so nearby values are likely to be on the same node (1). Strings use a random hash key (1) so there is no link between value and location(1). Hashing is more likely to give a uniform distribution over nodes that range based.

c) Indexed fields should have a wide range of possible values, ideally their values should be unique. You should also consider what fields might be searched on and what queries you should try to cover with the index. Indexes are held in memory, so indexing everything would probably fill memory. It might also slow down writes as every change to the DB would lead to a change to the index.

d) Map Reduce performs aggregation queries across a distributed database. The map function processes each document in turn and emits a key, value pair. The key is used to group the data and the value is passed to the reduce function for aggregation. The values are organised by key so the reduce function is given an array of values, all from the same key. The reduce function may be called more than once with intermediate sub-aggregations so its function must allow inputs in any order, must be idempotent (no need to use this word for the marks), i.e. reduce(key,[reduce(key, valuesArray)]) == reduce(key,valuesArray) and must produce the same result when given a mix of data and sub-aggregation results as it gives when given all of the data together, e.g. reduce(key, [C, reduce(key,[A, B])]) == reduce(key,[C, A, B]). Finalize allows a final pass over the completed results from reduce. 1 for map, 1 for reduce, 1 for finalize.

e) –

i. Relationships in a relational database do not have any associated fields or values. They simply link a foreign key to a primary key. In a graph database, relationships are also entities, meaning they have attributes.

ii. In a relational database, relationships are at the level of table to table, so all records in one table may refer to the same field in a home table. In a graph database, relationships are between entries and different entries can have different types of relationship.

Question 2

This question requires analytical thinking and design decisions to be made

a) –

i. The schema would need to define a field for every possible descriptor, this would be impossible.

ii. { “Manufacturer”:”Lenovo”, “Battery Life”:”Too short”, “Screen”:”Very Good”, “Liked”:[“Design”,”Colour”,”Weight”]} Strings should have quotes, curly braces and square brackets should be right.

iii. Searching is made more difficult if you don’t know the field names. Other good answers will also do.

b) –

i. Embedding would make the user documents potentially very long, but keep all their reviews in the same place. Finding all the reviews of a single person is a rare use case, so this is not the best design. A better choice would be to link from the user document to the review documents, allowing reviews of similar products to be kept together.

c) –

i. db.users.insert({username:”bill1”, Name:”Bill Smith”})

ii. db.users.update({username:”bill1”},{$set:{Age:25}})

iii. db.users.update({username:”bill1”},{$set:{email:[“bill1@bill.com”,”bill1@gmail.com”]}})

iv. db.users.update({username:”bill1”},{$push:{email:” bill1@work.com”}})

v. db.items.find({manufacturer:”Apple”})

vi. db.reviews.group(
key: {Category:1},
reduce: function(cur,av){av.n++; av.sum=av.sum+cur.rating;
av.av=av.sum/av.n;},
initial:{n:0, sum:0, av:0})