Big Data For Engineers

Gian Silvan Hiltbrunner

June 2020

Parts of the information provided within this document may be incomplete and/or incorrect. For corrections please submit a pull request to:
https://github.com/gianhiltbrunner/BigDataForEngineersSummary

Contents

1 Introduction

1.1 Are you capable of sketching the history of databases (ancient and modern) to a colleague in a few minutes?

Early forms of databases include natural forms of information storage such as speaking, singing and writing. These forms became more complex in time when people performed simple accounting tasks using stone inscriptions and later on when books became widespread with the invention of the book press. The computer then finally enabled us to store and manage data on much larger scales.

In the 1960s file systems were introduced allowing simple forms of data managment, in the 1970s relational table databases such as SQL were introduced allowing systematic queries. In the 2000s other databases became more popular (NoSQL), these include key-value stores, triple stores, document stores and column stores.

1.2 Do you know who Edgar Codd is?

Edgar Codd was an English computer scientist who invented the relational model for database management, the theoretical basis for relational databases.

1.3 Can you explain what Data Independence is, and why it is important?

The actual data and application programs acting on the data remain unimpaired even if the storage representation or access methods are changed. This is important when we want to store small scale data on a single hard drive and store large scale data on cluster infrastructure but essentially want to access the data in the same manner.

1.4 Do you know the rough conceptual difference between data, information, and knowledge?

- Data is conceived of as symbols or signs, representing stimuli or signals lacing any interpretation.
- Information is inferred from data", in the process of answering interrogative questions (e.g., "who", "what", "where", "how many", "when"), thereby making the data useful for "decisions and/or action".
- Knowledge represents information combined with understanding, capability and experience.

1.5 **Can you cite the five fundamental shapes of data, and how structured data can be characterized?**

1.6 **Can you briefly explain what a data model is?**

1.7 **Do you know the 8 standard prefixes of the International System of Units (when the exponent in base 10 is a positive multiple of 3)?**

1.8 **Can you list the main four technologies commonly referred to as NoSQL?**

1.9 **Do you know the three Vs?**

1.10 **Can you briefly define capacity, throughput and latency? Do you know their typical units?**

1.11 **Can you explain why and how the evolution of capacity, throughput and latency over the last few decades has influenced the design of modern databases?**

1.12 **Can you name a few big players in the industry that accumulate and analyze massive amounts of data?**

1.13 **Do you know the difference between bit and byte?**

1.14 **Can you name a few concrete examples that illustrate the various orders ofmagnitude of amounts of data?**

2 Lessons learnt

2.1 Can you explain why it is important to take into consideration whether a use case is read-intensive, or write-intensive, or in-between?

2.2 Can you explain why normal forms are important?

2.3 Can you describe the first normal form in simple terms?

2.4 Can you describe in simple terms (that is, without knowing the details of them) how higher normal forms (like Boyce-Codd) are related to joins?

2.5 Can you explain why it is common, for large amounts of data, to drop several levels of normal form, and denormalize data instead?

2.6 Can you give simple examples of denormalized data?

2.7 Can you explain what a declarative language is?

2.8 Can you explain what a functional language is?

2.9 Can you explain why it matters to design query languages that are declarative and functional?

2.10 Can you describe the major relational algebra operators: select, project, group and aggregate, join, Cartesian product, union, intersection, etc?

2.11 Do you know what each letter stands for in ACID and CAP?

2.12 Can you explain why, for large amounts of data, CAP becomes relevant over ACID?

2.13 Do you know the names of the basic components of the tabular shape at an abstract level (table, row, column, primary key) as well as the names of the most common corresponding counterparts in the NoSQL world?

2.14 Do you know the basic SQL constructs: SELECT FROM WHERE, GROUP, HAVING, JOIN, ORDER BY, LIMIT, OFFSET, as well as nested queries?

3 Object storage

3.1 Can you describe the limitations of traditional (local) file systems?

3.2 Can you explain what object storage is?

3.3 Can you explain what the benefits of object storage are?

3.4 Can you contrast or relate object storage with block storage, a file system, and the key-value model?

3.5 Can you explain the three different ways, on the physical level, to deal with more data (scale up, scale out, write better code)?

3.6 Can you explain why scaling out is less expensive than scaling up?

3.7 Can you explain what aspects of the design of object storage enable scaling out and why?

3.8 Do you know what a data centre is made of (racks, server nodes, storage nodes, etc.)?

3.9 Do you know the difference between storage, memory, CPU and network and how the three are paramount in a cluster?

3.10 Do you know rough, typical numbers (per-node storage capacity, memory, number of cores, etc.)?

3.11 Do you know how storage and memory technologies (HDD, SDD and RAM) compare in terms of capacity and throughput?

3.12 Can you sketch the (key-value) data model behind object storage? Can you spot instances of the key-value model in several other technologies seen in the lecture?

3.13 Do you know the difference between data and metadata?

3.14 Do you know the order of magnitude that can be achieved in terms of number of objects, and object size?

3.15 Can you name a few big players for cloud-based object storage (vendors, consumers)?

3.16 Can you describe the features of S3 and Azure Blob Storage on a high level? Do you know what a bucket and object are? What block blob storage, append blob storage and page blob storage are and how they work?

3.17 Can you describe what the most important SLA (Service Level Agreement) parameters mean (e.g., latency, availability, durability) as well as their typical range?

3.18 Can you describe a typical use case for object storage?

3.19 Can you explain what a REST API is, what resources and methods are?

3.20 Can you explain how an object storage API looks like (get and put)?

3.21 Are you able to upload and download objects from S3 or Azure Blob Storage?

4 Distributed file systems

4.1 Can you explain the difference between block storage and object storage?

4.2 Can you explain the difference between the (logical) key-value model and a file system?

4.3 Can you contrast block storage and object storage in terms of maximum number of objects/files?

4.4 Can you contrast block storage and object storage in terms of the maximum size of objects/files?

4.5 Do you know the order of magnitude of a block size for a local filesystem and for a distributed file system? Can you explain the rationale behind them with respect to latency and throughput?

4.6 Do you know where HDFS shines and why?

4.7 Do you know that HDFS files are updated by appending atomically and why?

4.8 Do you know how HDFS performs in terms of throughput and latency?

4.9 Do you know the main benefits of HDFS (commodity hardware, etc.)?

4.10 Can you contrast master-slave architectures to peer-to-peer architectures?

4.11 Can you explain the HDFS architecture, what a namenode is and what a datanode is, how blocks are replicated?

4.12 Can you sketch how the various components communicate with each other (client, namenode, datanode)?

4.13 Can you point to the single points of failure of HDFS and explain how they can be addressed?

4.14 Can you explain how the namenode stores the file system namespace, in memory and on disk? In particular, can you explain how the namespace file and the edit log work together at startup time and how they get modified once the system is up and running?

4.15 Can you explain what a standby namenode is, what it is and what it is not?

4.16 Are you able to use the HDFS shell commands (creating directories, reading, uploading and downloading files, etc.)?

5 Syntax

5.1 Can you describe why syntax is relevant to data management and to Big Data, for all data shapes?

5.2 Can you give examples of syntax for trees? For tables? (The two data shapes extensively covered in the lecture)

5.3 Can you explain what well-formedness is with respect to syntax?

5.4 Can you list the XML basic building blocks (we covered document, element, attribute, text) and do you know what they look like?

5.5 Can you tell whether a given XML document is well-formed, with a software (oXygen) as well as with your own eyes?

5.6 Do you know the five fundamental pre-defined entities (used to escape characters: & < > ' ") and their syntax?

5.7 Do you know when some characters must be escaped (e.g., < in text)?

5.8 Can you list the JSON basic building blocks (object, array, string, number, boolean,null) and do you know what they look like?

5.9 Can you tell whether a given JSON document is well-formed, also with your owneyes?

6 Wide column stores

6.1 Can you explain the limitations of the traditional relational model?

6.2 Can you explain the differences and similarities between a wide column store andthe traditional relational model?

6.3 Can you explain why wide column stores are called wide column stores, in particularon the storage level?

6.4 Do you know and can you contrast the two ways data can be distributed(partitioning, replicating)?

6.5 Can you explain the data model behind wide column stores, in particular, rows,columns, column families and cells?

6.6 Can you explain the motivation behind data getting denormalized into more columnfamilies?

6.7 Can you explain what aspects of an instance (table) in a column store must betypically known in advance, and which can be changed on the fly?

6.8 Can you name a few big players, in particular the one behind the initial founding paper?

6.9 Can you explain why HBase is based on HDFS, yet low-latency?

6.10 Can you explain what regions are for wide column stores?

6.11 Do you know how to identify a region based on the content of the wide columnstore?

6.12 Do you know the four basic kinds of (low-level) queries in HBase?

6.13 Can you describe the physical architecture of a wide column store like HBase, andcompare it with that of a distributed file system like HDFS?

6.14 Do you know the physical layers of HBase (table, region, Store, Memstore, HFile,HBase block, KeyValue)?

6.15 Do you know the difference between an HDFS block and an HBase block as well astheir typical sizes?

6.16 Can you explain how new cells are written to HBase via the cell store?

6.17 Can you explain how cells are read from HBase via both the stored HFiles and theMemstore?

6.18 Can you explain what compaction and flushing is?

6.19 Can you sketch the contents of an HFile and know what it corresponds to in anHBase table?

6.20 Can you explain why it is crucial not to have too long column family names?

6.21 Can you populate data into, and execute queries on top of HBase?

7 Data models

7.1 Can you explain the difference between syntax and a data model?

7.2 Can you explain why and how trees can be used to model data that is denormalized(i.e., not in first normal form)?

7.3 Can you relate syntaxes to data models (e.g., CSV to tables, XML/JSON to trees, ...)?

7.4 Can you explain why trees modeling XML (infoset) have labels on the nodes, whiletrees modeling JSON have labels on edges?

7.5 Can you name a few data models for XML (Infoset, PSVI, XDM)?

7.6 Are you able, given an XML document, to sketch a tree representing that data (witha document node, elements, attributes, text, ...)?

7.7 Do you know the different between an atomic type/value, and a structured-type/value (regardless of the exact data model)?

7.8 Can you name a few atomic types found across a broad number of data models?

7.9 Do you know the difference between the lexical space and the value space of anatomic type?

7.10 Can you tell the difference between structured types based on lists (e.g., JSON array) and maps (e.g., JSON object)?

7.11 Can you give examples of type cardinalities, and their associated symbols?

7.12 Can you explain the difference between well-formedness and validity? Do you knowwhat extra information you need to assess validity?

7.13 Are you able to design simple XML Schemas to validate XML? Can you restrict simpletypes to allow specific values or value fulfilling specific criteria (length, pattern...)? Can you explain how to declare elements with simple types? Can you explain how to declare elements with complex types and various content models (empty, simplecontent, complex content, mixed content)?

7.14 Can you tell, given an XML Schema, whether an XML document is valid against it,with software (oXygen) but also with your own eyes?

7.15 Are you able to design simple JSON Schemas to validate JSON?

7.16 Can you name (without details) further data modeling technologies for tree-likedata? (Avro, protocol buffers, ...)

7.17 Do you understand that a table in first normal form can be seen as a homogeneouscollection of flat trees (one per row and all with the same attributes)?

8 Massive parallel processing (MapReduce)

8.1 Can you explain the patterns that appear in large-scale data processing: map, shuffle?

8.2 Can you explain how the mastery of these patterns can bring significant improvements in performance?

8.3 Can you explain the MapReduce model, also in terms of the patterns depicted above (map, shuffle, reduce)?

8.4 Can you describe the physical architecture of MapReduce (map tasks, reduce tasks as well as data flow)? Can you explain version 1 of the architecture (JobTracker, TaskTrackers) and its limits?

8.5 Can you outline what a map function looks like, and what a reduce function looks like?

8.6 Do you know the main input and output types of MapReduce?

8.7 Can you explain how combining improve MapReduce's performance?

8.8 Can you state the assumptions behind reusing the reduce function as a combinefunction?

8.9 Do you understand, in some simple cases, how to design a combine function thatwould make a MapReduce job faster, even if the combine function is not the exact same as the reduce function? (Example: computing an average, which requires keeping track of the weights in the output of the combine function).

8.10 Can you explain why MapReduce shines on top of a distributed file system: "Bring the query to the data"?

8.11 Can you explain why MapReduce especially makes sense when the bottleneck is the speed of reading and writing data from the disk (as opposed to other bottlenecks such as storage capacity or CPU usage)?

8.12 Can you explain how MapReduce splits differ from HDFS blocks, what impedance mismatches arise and how they are addressed?

8.13 Do you know what the Java API of MapReduce looks like on a high level (Version 2 of the API that we covered, not to be confused with Version 2 of MapReduce running on YARN)?

9 Resource Management

9.1 Can you explain how YARN works, and how it can be used to improve MapReduce, and to support other technologies like Spark?

9.2 Can you describe the YARN components?

9.3 Do you know what a ResourceManager does?

9.4 Do you know what a NodeManager does?

9.5 Do you know what and where a Container is?

9.6 Do you know what an ApplicationMaster is and does?

9.7 Can you list the main resources that are managed in a cluster? (Disk storage,memory, CPU, network I/O).

9.8 Can you explain, in simpler words, what the added value of YARN is? Can you explainwhat it is an improvement over the first version of MapReduce, which was taking care of resource management on its own, and had issues with this?

10 Massive parallel processing (Spark)

10.1 Can you explain how Spark is more powerful than MapReduce on a data model level?

10.2 Can you explain what a Resilient Distributed Dataset (RDD) is?

10.3 Do you know the difference between an action and a transformation?

10.4 Do you know the main actions and transformations available in Spark?

10.5 Are you able to classify transformations and actions in various buckets (those thatwork on any value, those that work on key-value pairs, unary binary transformations, ...)?

10.6 Can you describe how transformations run physically (tasks, stages...)?

10.7 Can you explain when and how a series of transformations can be optimized bykeeping the same set of machines with no network communication between the transformations?

10.8 Can you explain what a stage is? Can you relate it to transformations? To tasks? ToJobs?

10.9 Can you tell why and when shuffling is needed? In other words, can say whether atransformation has a narrow dependency or a wide dependency?

10.10 Can you easily draw a directly acyclic graph for a Spark job, and mark the stages?

10.11 Do you understand why keeping an RDD persisted can be useful and improveperformance?

10.12 Can you explain how controlling the way data is partitioned can make executionfaster because we can influence stages?

10.13 Can you explain what a DataFrame is and what its benefits are?

10.14 Can you describe the limitations of DataFrames?

10.15 Can you write queries on DataFrames, both in Spark SQL or using DataFrames-transformations?

11 Document stores

11.1 Do you understand how collections in documents store generalize the concept of a relational table?

11.2 Can you explain what documents store can do that relational databases cannot (e.g., heterogeneous collections, schema-less collections, data denormalized into trees...)?

11.3 Do you know how to issue MongoDB queries on a low level? Do you know the parameters of the find() function? (query, then projection, then sorting)?

11.4 Can you explain how, in a document store, the documents can be sharded and replicated?

11.5 Do you understand how indices can make queries faster, like in relational databases?

11.6 Do you know what kinds of indices there are (hash, B-trees)?

11.7 Are you capable of telling if an index is useful to a given query, for simple settings(for example, an index on a single field and a query that selects on that field)?

11.8 Do you know that a compound index (e.g., on keys a and b) can also be used as anindex on any prefix of the compound key (e.g., on key a only) "for free"?

12 Querying trees

12.1 Can you explain why a language such as JSONiq provides, in the context of document stores, exactly the same functionality as SQL in a relational database?

12.2 Can you name the first-class citizen of the JDM (sequence of items)?

12.3 Can you name various kinds of items in the JDM?

12.4 Can you name a few query languages in the XML/JSON ecosystem?

12.5 Are you able, in JSONiq, to construct items (atomic items, elements, etc.)?

12.6 Are you able, in JSONiq, to perform logical operations? Do you understand what theEffective Boolean Value of a sequence and how it relates to logical operations?

12.7 Are you able, in JSONiq, to perform arithmetic operations (addition, etc.)? Do youunderstand the constraints on the input sequences of such operations? Can you explain the behavior of these operations on empty sequences? Can you explain what happens if one of the two operands is a node and not an atomic item?

12.8 Are you able, in JSONiq, to perform comparisons (lt, ge, etc.)? Do you understand the constraints on the input sequences of such operations? Can you explain the behavior of these operations on empty sequences? Can you explain what happens if one of the two operands is a node and not an atomic item?

12.9 Do you understand how general comparisons (<, >=, etc.) work on sequences with more than one item, and implicitly use an existential quantifier)?

12.10 Do you understand how FLWOR expressions work and describe what they return? (for clause, let clause, where clause, order by clause, etc.)

12.11 Are you able to use further expressions (if-then-else, switch, ...)?

12.12 Do you understand how to dynamically build JSON content with object and arrayconstructors?

12.13 Do you understand that expressions can be combined at will, as any expression takesand returns sequences of items? Do you know how to use parentheses to make precedence clear, like you did in primary school with + and *?

12.14 Do you know the JSONiq type syntax (atomic types taken from XML Schema, syntaxfor object and array types, as well as cardinality symbols), and how to use typechecking (instance of, cast as, etc.)?

12.15 Given a collection of JSON objects (for example JSON Lines on HDFS), are you able touse Rumble to write JSONiq queries (FLWOR) that do projection? selection? (weleave grouping, join, ... aside in the context of this lecture, however you need to know that this can be done with JSONiq as well).