Question 1: One way of classifying data is into the categories: structured, semi-structured and unstructured. Give a brief description of each type and an example of each type.

Answer:

1. Structured data resides in predefined formats and models. Structured data is generally tabular data that is represented by columns and rows in a database.

Examples: One example of structured data is a relational database such as a table.

2. Unstructured data is stored in its natural format until it's extracted for analysis.Unstructured data is information that either does not organize in a pre-defined manner or not have a pre-defined data model. Unstructured information is a set of text-heavy but may contain data such as numbers, dates, and facts as well.

Examples: Videos, audio, and binary data files might not have a specific structure. They're assigned to as unstructured data.

3. Semi-structured data basically is a mix of both structured and unstructured data. Semi-structured data is information that doesn't consist of Structured data (relational database) but still has some structure to it.

Examples: A few examples of semi-structured data sources are emails, XML and other markup languages, binary executables, TCP/IP packets, zipped files, data integrated from different sources, and web pages.

Cited from:
https://k21academy.com/microsoft-azure/dp-900/structured-data-vs-unstructured-data-vs-semi-structured-data/

https://treehousetechgroup.com/what-is-semi-structured-data-5-key-things-to-know/

Question 2: What is the difference between a type, e.g. VARCHAR, INT, and a *domain?* Given an example.

Answer:

A domain allows you to define a data type and re-use it across your data model. A domain describes the technical attributes of a field, such as the data type or the number of positions in a field. The domain defines primarily a value range describing the valid data values for the fields referring to this domain.

A data type is an atomic (lowest level) definition of a table column. It is also the same but where domain can be used for more than one data element, where as data type cannot.

Example: We can create domain like

```
Create domain person_name_type as varchar(40);
```

And use EITHER a domain OR a data type to define what the table consists of.

```
Create table a(
First person_name_type,
Last person_name_type);

Create table a(
First varchar(40),
Last varchar(40));
```

Cited from:
https://www.quora.com/What-is-the-difference-between-domain-and-data-type-in-DBMS
https://answers.sap.com/questions/2480018/difference-of-domain-and-data-types.html

Question 3: Assume that you know that access to a relational table will be primarily sequential. How would you lay the blocks out on a disk's cylinders, heads and sectors?

Answer:

Traditional disk data layout minimizes seek time and rotational latency. If access to a relational table will be primarily sequential. Then I can map a sequence of blocks on the same cylinder and sectors in order, such as cylinder i, sector j; cylinder i, sector j+1. Once a cylinder is full, do the same thing in the adjacent cylinder.

Cited from:
https://www.usenix.org/legacy/publications/library/proceedings/fast03/tech/full_papers/dramaliev/dramaliev_html/node6.html

Question 4: Most databases do not support hash indexes. What are three benefits of B+ Tree indexes over hash indexes? Express your answer by giving examples of SQL queries better supported by B+ Trees.

Answer:

1. B+ Trees is more efficient in a range query retrieval while the hash index is useless , because the original orderly key value, after the hash algorithm, may become discontinuous, there is no way to use the index to complete the scope Query retrieval.
2. In the same way, the hash index can't use the index to complete the sorting, and the partial fuzzy query like 'xxx%' (this partial fuzzy query is actually a range query in essence).
3. The keyword retrieval efficiency of the B+ tree index is better than hash indexes. In B+ trees, numerous keys can easily be placed on the page of memory because they do not have the data associated with the interior nodes. Therefore, it will quickly access tree data that is on the leaf node. In the case of a large number of repeated key values, the efficiency of the hash index is extremely low because there is a so-called hash collision problem.

Cited from:
https://medium.com/@mena.meseha/what-is-the-difference-between-mysql-innodb-b-tree-index-and-hash-index-ed8f2ce66d69
https://www.guru99.com/introduction-b-plus-tree.html

Question 5: What is the primary benefit of RAID-5 compared to RAID-0 and RAID-1?

Answer:

RAID level 5 uses disk striping and parity to strip data across three or more drives. RAID-5 provides both availability and improved performance, but incurs 20% storage overhead compared to 100% for RAID-0 and RAID-1.

Cited from:
https://platinumdatarecovery.com/blog/raid-comparison

Question 6: Briefly explain the difference between fixed size records and variable size records.

Answer:

A fixed size variable is one whose record size is fixed, like their length never varies, whereas a variable one is whose record size varies with changes in inputs. The fixed

size ensures the fixed size is never variable. A variable size is one in which the size of each record can vary, depending on what data is stored in the field.

Cited from:
https://brainly.in/question/20990182

Question 7: The database query optimizer may create a hash index to optimize a query. Give an example of a SELECT statement for which the optimizer may create a hash index.

Answer:

SELECT DISTINCT

Question 8: Least-recently-used is a common buffer pool replacement policy. But, sometimes LRU is the worst possible algorithm. Briefly give an example of when LRU is the worst algorithm and why.

Answer:

Nested loop join.

The query processor repeatedly iterates through the blocks of the permission/probe table. When moving from block N to N+1, block N is the most recently used and therefore the last choice for replacement. However, block N+1 is the block with the longest next access time, so it is the best block to reclaim. In a nested loop scan using LRU, the algorithm replaces blocks in the worst order.

Question 9: Briefly explain the concept of conflict serializable and how it differs from serializable.

Answer:

A schedule is called conflict serializable if we can convert it into a serial schedule after swapping its non-conflicting operations.

Difference: Conflict serializability is a subset of serializability. If a schedule is conflict serializable, it is implied that this schedules is serializable.

It is computationally easier to determine if something is conflict serializable as opposed to just serializable. Simply construct a precedence graph. If the graph is non-cyclic, then this schuedule is conflict-equivalent to some serial schedule described by the pathing of the graph.

Cited from:
https://www.geeksforgeeks.org/conflict-serializability-in-dbms/
https://stackoverflow.com/questions/20529800/whats-the-difference-between-conflict-serializable-and-serializable

Question 10: What is a deadlock? How do DBMS transaction managers break deadlock?

Answer:

In a database, a deadlock is an unwanted situation in which two or more transactions are waiting indefinitely for one another to give up locks. Deadlock is said to be one of the most feared complications in DBMS as it brings the whole system to a Halt.

The database manager rolls back uncommitted transactions from the selected process automatically. When the rollback operation is complete, locks that belonged to the victim process are released, and the other processes involved in the deadlock can continue.

Cited from:
https://www.geeksforgeeks.org/deadlock-in-dbms/
https://www.ibm.com/docs/en/db2/11.5?topic=management-deadlocks