Multiple Choice Question (MCQ) from Source Document

[BL-3: Applying]

Question 1:

A critical financial application requires constant, rapid access to individual, non-contiguous data records from a massive dataset. If this dataset is primarily stored on Network Storage, applying the principles of storage hierarchy and access times from the text, how would this affect the application's real-time performance compared to if the data were primarily in DRAM?

Options:

A) Performance would be negligibly different, as both are non-volatile storage.

B) The application would experience significantly degraded performance due to Network Storage's much higher latency and the inefficiency of random access on non-volatile storage.

C) Performance would improve, as Network Storage offers greater capacity.

D) The system would automatically migrate the data to faster storage without performance impact.

Correct Answer:

B

[BL-3: Applying]

Question 2:

Consider a disk-oriented DBMS where an Execution Engine needs to retrieve and then update "Page #3" which is currently stored on disk and not in the Buffer Pool. Applying the described DBMS architecture, what is the logical sequence of operations the DBMS would perform to handle this request?

Options:

A) The Execution Engine directly accesses Page #3 on disk, interprets its layout, updates it, and then signals the Buffer Pool to load the updated page.

B) The Execution Engine requests Page #3; the Buffer Pool fetches it from disk, provides a pointer; the Engine interprets and updates; the Buffer Pool then notes the update for eventual write-back.

C) The Directory first updates its record for Page #3, then the Execution Engine accesses it, updates it, and finally loads it into the Buffer Pool.

D) The DBMS bypasses the Buffer Pool for updates, directly modifying Page #3 on disk, which is then eventually loaded into the Buffer Pool for consistency.

Correct Answer:

B

[BL-3: Applying]

Question 3:

A database administrator is designing a system for storing sensor data that arrives in a continuous stream. The primary requirements are high-throughput insertion and the ability to iterate through all recent data, without a strict need for the data to be sorted or indexed within the file itself. Based on the file storage architectures presented, which organization would be most appropriate for these requirements?

Options:

A) Tree File Organization, because it ensures data is sorted, which is optimal for all retrieval.

B) Sequential / Sorted File Organization (ISAM), as it guarantees ordered data for iteration.

C) Heap File Organization, because it's an unordered collection of pages allowing efficient append-only insertions and supports iteration over all pages.

D) Hashing File Organization, as it provides the fastest direct lookups by hash.

Correct Answer:

C

[BL-3: Applying]

Question 4:

A database table is defined with the schema CREATE TABLE sensor_data (id INT, reading_time TIMESTAMP, status CHAR(1), location_id INT);. To ensure all attributes are word-aligned for optimal CPU access, as discussed in the text, how would a DBMS typically arrange the status (CHAR(1)) and location_id (INT) attributes relative to id (INT) and reading_time (TIMESTAMP) within the tuple's physical layout, assuming a 64-bit word architecture?

Options:

A) The DBMS would always store attributes in the order specified in the CREATE TABLE statement and rely solely on padding after each attribute to achieve alignment.

B) The DBMS would attempt to reorder the attributes, placing smaller, word-sized attributes like INT and CHAR(1) together or after larger attributes like TIMESTAMP, potentially followed by minimal padding, to fit them efficiently within 64-bit words.

C) The DBMS would convert CHAR(1) and INT to larger data types that inherently fit 64-bit words, irrespective of their original definition.

D) The DBMS would store all attributes unaligned, as modern CPUs can handle unaligned access without performance penalty.

Correct Answer:

B

[BL-3: Applying]

Question 5:

A software engineer is designing a database schema for a new banking application. They need to store account balances which require absolute precision and also account for instances where an optional comment field might be empty. Based on the discussed data types and NULL handling strategies for row-stores, what would be the most appropriate choices for these requirements?

Options:

A) Use FLOAT for account balances due to its speed, and "Per Attribute Null Flag" for the comment field for simplicity.

B) Use NUMERIC or DECIMAL for account balances to guarantee arbitrary precision, and the "Null Column Bitmap Header" approach for the comment field as it is common in row-stores.

C) Use INT for account balances (scaled to cents) and "Special Values" (e.g., empty string) for the comment field.

D) Use DOUBLE for account balances for broad range, and "Per Attribute Null Flag" to clearly indicate NULLs in the comment field.

Correct Answer:

B