| Register Number | | | | | | | | | |
|---|---|---|---|---|---|---|---|---|---|

**Sri Sivasubramaniya Nadar College of Engineering, Kalavakkam – 603 110**

(An Autonomous Institution, Affiliated to Anna University, Chennai)

## Department of Computer Science and Engineering

**Continuous Assessment Test – III**

**Question Paper**

| **Degree & Branch** | B.E CSE | | | | **Semester** | IV |
|---|---|---|---|---|---|---|
| **Subject Code & Name** | UCS1404 Database Management Systems | | | | **Regulation: 2018** | |
| **Academic Year** | 2021-22 | **Batch** | 2020-24 | **Date** | **31.05.2022** | **FN** |
| **Time: 8:30 -10:00 am** | **Answer All Questions** | | | | **Maximum: 50 Marks** | |

**Part – A (6×2 = 12 Marks)**

| | | | |
|---|---|---|---|
| <K2> | 1. Compare shared lock and exclusive lock.<br>Shared lock can be placed on objects that do not have an exclusive lock already placed on them. Exclusive lock can only be placed on objects that do no have any other kind of lock | <CO4> | 2.1.2 |
| <K3> | 2. Consider the following schedule: **Sa: w1(X); w2(X); a1; Categorize** Sa based on recoverability. Not a strict schedule | <CO4> | 2.2.3 |
| <K3> | 2. If a transaction begins in a database and committed, and later the transaction fails for some reason whether the database can be restored to the previous state. Identify the reasoning behind it.<br>**It can restored using Redo of operations by looking into log file.** | <CO4> | 2.2.3 |
| <K3> | 4. Consider the schema R1 with attributes (id, product, itemsInStock) with the values (1,"iPhone",10). If the following transactions were in place.<br><br>**T1** / **T2**<br>Updates ItemsInstock : 9<br>Billing Customer<br>Reads ItemsInstock : 9<br>Insufficient Funds<br>Transaction Rolled back<br>Reverts ItemsInstock : 10<br><br>Identify the type of the concurrency control problem the transactions are facing . How it can be avoided?<br>**Dirty Read problem. Avoided by reading from a committed transactions** | <CO4> | 2.2.3 |
| <K1> | 5. List the categories of NOSQL systems<br>Document based, Key Value store, Columnar based and Graph based | <CO5> | 1.4.1 |

| | | | |
|---|---|---|---|
| <K3> | 6. Identify the appropriate MongoDB command to find the documents having employee salary greater than 10,000 from employee collections.<br><br>**db.Employee.find({salary : {$gt:10000}}).forEach(printjson);** | <CO5> | 2.2.3 |

**Part – B (3×6 = 18 Marks)**

| | | | |
|---|---|---|---|
| <K2> | 7. During execution, a transaction passes through several states, until it finally commits or aborts. Summarize all possible sequences of states through which a transaction may pass through state transistion diagram. Outline why each state transition should occur,<br><br>A transaction is an atomic unit of work that is either completed in its entirety or not done at all.<br>Transaction states: **Active state:** Transaction goes into active state immediately after it starts execution.<br>**Partially Committed State:** When the transaction ends it moves to the partially committed state.<br>**Committed State:** If the transaction reaches its commit point, the transaction enters the committed state.<br>**Failed state:** If the transaction is aborted during its active state, it enters into the failed state.<br>**Terminated State:** Corresponds to the transaction leaving the system. | <CO4> | 1.4.1<br>2.1.2 |
| <K3> | 8. Amount of $100 has been deducted from account holder "X" and added to the account holder "Y". Make use of the example to discuss<br>"ACID" properties of the transactions and explain the same by defining the "ACID" properties.<br>Transaction should possess several properties often called as the ACID properties that should be enforced by the concurrency control and recovery methods.<br>Atomicity:A transaction is an atomic unit of processing; It is either performed in its entirety or not performed at all. It is the responsibility of the transaction recovery subsystem. Consistency preservation: Every transaction should execute from beginning to end without interference of other transactions. A correct execution of the transaction must take the database from one consistent state to another. It is the responsibility of the programmer or the DBMS module that enforces integrity constraints.<br><br>Isolation: A transaction should not make its updates visible to other transactions until it is committed. This property, when enforced strictly, solves the temporary update problem and elminates cascading rollbacks. Level 0: no dirty read Level 1: no lost updates Level 2: no lost updates and no dirty read Level 3: Level 2 + repeatable reads It is enforced by concurrency control subsystem of the DBMS Durability or permanency: Once a transaction changes the database and the changes are committed, these changes must never be lost because of subsequent failure. It is the responsibility of the recovery subsystem of the DBMS . Recovery protocols enforces atomicity and durability | <CO4> | 2.1.2<br>2.2.3 |

| | | | |
|---|---|---|---|
| | Transfer 50 rupees from account a to account b read(a); a:=a-50; write(a); read(b); b=b+50; write(b)<br>Atomicity: a must not be debited without crediting b Consistency: Sum of a and b remains constant Isolation : Account a is making T1 and T2 transactions to account b and c, but both are executing independently without affecting each other. It is known as Isolation. Durability: if b is notified of credit, it must persist even if the database crashes | | |
| <K2> | 9. Explain CAP theorem with an example<br>The CAP theorem (also called Brewer's theorem) states that a distributed database system can only guarantee two out of these three characteristics: Consistency, Availability, and Partition Tolerance. A system is said to be consistent if all nodes see the same data at the same time.<br>Three letters in CAP refer to three desirable properties of distributed systems with replicated data<br>Consistency: The nodes will have same copies of a replicated data item visible for various transactions.<br>Availability: Guarantees that every request receives a response about whether it was successful or failed. Partition<br>Tolerance: System can continues to operate despite communication breakages that separate the cluster into partitions, where the nodes in each partition can only communicate among each other. | <CO5> | 1.4.1<br>2.2.3 |

**Part – C (2×10 = 20 Marks)**

| | | | |
|---|---|---|---|
| <K3> | 3. $S_1$: $r_1(X)$, $r_3(Y)$, $\mathbf{r_2(X)}$, $\mathbf{w_1(X)}$, $w_2(X)$, $w_3(Y)$, $r_1(Z)$, $w_1(Z)$, $r_3(Z)$, $w_3(Z)$<br>a) Draw the precedence graph for $S_1$ and state whether the schedule is serializable or not. Why or why not? (2)<br>T1 T2<br>T3<br>Not serializable, as there is a cycle between T1 and T2.<br>*b)* Now swap the operations in $S_1$ that is highlighted in bold and consid<br>Check whether the schedule S2 is conflict serializable through the swapping of operations. If so, give the equivalent serial schedule(s). | | 1.4.1<br>2.1.2<br>2.2.3 |

| T1 | T2 | T3 |
|---|---|---|
| r(X) | | |
| | | r(Y) |
| w(X) | | |
| | r(X) | |
| | w(X) | |
| | | w(Y) |
| r(Z) | | |
| w(Z) | | |
| | | r(Z) |
| | | w(Z) |

After swapping non-conflicting operations:

| T1 | T2 | T3 |
|---|---|---|
| r(X) | | |
| w(X) | | |
| r(Z) | | |
| w(Z) | | |
| | r(X) | |
| | w(X) | |
| | | r(Y) |
| | | w(Y) |
| | | r(Z) |
| | | w(Z) |

The equivalent serial schedule: T1 → T2 → T3

| T1 | T2 | T3 |
|---|---|---|
| r(X) | | |
| w(X) | | |
| r(Z) | | |
| w(Z) | | |
| | | r(Y) |
| | | w(Y) |
| | | r(Z) |
| | | w(Z) |
| | r(X) | |
| | w(X) | |

| | | | |
|---|---|---|---|
| | c) Is the schedule $S_2$ is view serializable or not? Justify. If so, determine the equivalent serial schedule(s).<br><br>(4)<br>Justification of S2 as view serializable .<br><br>For schedule S2:<br>View of X: T1 writes →<br>T2 reads view of Z: T1<br>writes → T3 reads Last<br>Write(X): T2<br>Last Write(Z): T3<br><br>For serial: T1 → T2 → T3<br>View of X: T1 writes →<br>T2 reads view of Z: T1<br>writes → T3 reads Last<br>Write(X): T2<br>Last Write(Z): T3<br>Hence S2 is view serializable to serial T1 → T2 → T3<br><br>For serial: T1 → T3 → T2<br>View of X: T1 writes →<br>T2 reads view of Z: T1<br>writes → T3 reads Last<br>Write(X): T2<br>Last Write(Z): T3<br>Hence S2 is view serializable to serial T1 → T3 → T2 | | |
| | (OR) | | |
| <K3> | 11. Consider the two transactions T1 and T2 :<br><br>**T1 : r1 (Y); r1 (X); w1 (X);**<br>**T2 : r2 (X); w2 (X);**<br><br>Assume that the schedule to be generated from the above transactions must use two-phase locking protocol (2PL) using shared / exclusive locks. The schedule does not allow upgradation / degradation of locks.<br>a) Without changing the order of operations in the transaction, write a serializable schedule that follows basic 2PL.<br>b) Now consider the modified version of transaction T2 as T3:<br>**T3: r2(X); r2(Y); w2(X);** and add the operation commit at the end of each transaction T1 and T2.<br>Write a serializable schedule that implements strict 2PL.<br><br>Serializable schedule will have interleaving of operations. Basic 2PL has growing and shrinking phase in locking.<br>*Assumption*: No upgradation/downgradation of locks. Uses shared/exclusive locks. | <CO4> | 1.4.1<br>2.1.2<br>2.2.3 |

| | | | |
|---|---|---|---|
| | This is the only possible serializable schedule using 2PL<br><br>**T1** \| **T2**<br>*S_lock(Y)* \|<br>*r(Y)* \|<br>*Unlock(Y)* \|<br>\| *X_lock(X)*<br>\| r(X)<br>\| w(X)<br>\| *Unlock (X)*<br>*X_lock(X)* \|<br>*r(X)* \|<br>*w(X)* \|<br>*Unlock(X)* \|<br><br>a) Can you have a strict 2PL schedule with out the *commit/abort* operation? Why or why not?<br>Now consider the modified version of transaction $T_2$ *as* $T_2$'' :<br>$T_2$ : $r_2(X)$; $r_2(Y)$; $w_2(X)$; and add the operation *commit* at the end of each transaction $T_1$ and $T_2$ . Write a serializable schedule that implements *strict* 2PL.<br><br>Strict 2PL releases the **write locks** only after reaching the termination (*commit / abort*) of transaction.<br><br>**T1** \| **T2''**<br>*S_lock(Y)* \|<br>*r(Y)* \|<br>\| *X_lock(X)*<br>\| *r(X)*<br>\| *S_lock(Y)*<br>\| *r(Y)*<br>\| w(X)<br>\| ***Commit;***<br>*X_lock(X)* \|<br>*r(X)* \|<br>*w(X)* \|<br>***Commit;*** \| | | |
| <K2> | 12. Outline the characteristics of NOSQL systems with respect to distributed databases, data models and query language.<br><br>Not requiring a Schema: Semi-structured and self-describing data of NOSQL systems does not requires a schema. Partial schema was allowed to improve storage efficiency JSON is used in several NOSQL systems Less powerful Query languages: NOSQL system provide a set of functions and operations as a programming API. Reading and writing of data objects are accomplished by calling appropriate operations by the | <CO5> | 1.4.1<br>2.2.3 |

| | | | |
|---|---|---|---|
| | programmer. CRUD operations : for Create, Read, Update, and Delete. SCRUD operations because of an added Search (or Find) operation Versioning: Some NOSQL systems provide storage of multiple versions of the data items, with the timestamps of when the data version was created. | | |
| | | | |
| <K2> | 13. Outline the Data model, CRUD operations and Distributed system characteristics of document based NOSQL Systems, | <CO5> | 1.4.1 2.2.3 |

---

| Prepared By | Reviewed By | Approved By |
|---|---|---|
| Course Coordinator | PAC Team | HOD |

**Guidelines**

1. The question paper should be set in accordance with Bloom's Taxonomy (APPENDIX – A: next page) . The questions in a desired knowledge level must contain the respective action verbs.
2. The Knowledge level (Eg. <K2>), course outcome (Eg. <CO2>), and the program indicators (Eg. <1.2.1>) should be mentioned against each question and subdivisions in the respective columns.
3. Both the questions in "either or" type must be set in the same knowledge level and must be from the same CO.
4. In the case of "either or" type questions, the keyword (OR) must be in a separate row.
5. In the case of sub-divisions in a question, it is preferable to have the same knowledge level.
6. The marks assigned to each question in the case of subdivisions should be mentioned clearly at the end of the question within brackets and with a keyword Marks. (Eg. (5 Marks) ).
7. Add the keyword "Options" before the choices of an objective type question in Part A.
8. Once the question paper is set, its adherence to the guidelines in terms of knowledge levels and marks distribution has to be approved by the QP Scrutiny Team.

## APPENDIX – A
## Bloom's Taxonomy Action Verbs

| K Level | Bloom's Definition | Action Verbs |
|---|---|---|
| K1 Remember | Exhibit memory of Previously learned Material by recalling facts, terms, basic concepts, and answers. | Choose, Define, Find, How, Label, List, Match, Name, Omit, Recall, Relate, Show, Spell, Tell, What, When, Where, Which, Who, Why. |
| K2 Understand | Demonstrate understanding of facts and ideas by organizing, comparing, translating, interpreting, giving descriptions and stating main ideas. | Classify, Compare, Contrast, Demonstrate, Explain, Extend, Illustrate, Infer, Interpret, Outline, Relate, Rephrase, Show, Summarize, Translate |
| K3 Apply | Solve problems to new situations by applying acquired knowledge, facts, techniques, and rules in a different way. | Apply, Build, Construct, Develop, Experiment with, Identify, Interview, Make use of, Model, Organize, Plan, Select, Solve, Utilize |
| K4 Analyse | Examine and break information into parts by identifying motives or causes. Make inferences and find evidence to support generalizations | Analyze, Assume, Categorize, Conclusion, Discover, Dissect, Distinguish, Divide, Examine, Function, Inference, Inspect, Motive, Relationships, Simplify, Survey, Take part in, Test for, Theme |
| K5 Evaluate | Present and defend opinions by making judgments about | Agree, Appraise, Assess, Award, Choose, Compare, Conclude, Criteria, Criticize, Decide, Deduct, Defend, Determine, |

| | | |
|---|---|---|
| | information, validity of ideas, or quality of work based on a set of criteria. | Disprove, Estimate, Evaluate, Explain, Importance, Influence, Interpret, Judge, Justify, Mark, Measure, Opinion, Perceive, Prioritize, Prove, Rate, Recommend, Rule on, Select, Support, Value |
| K6<br>Create | Compile information together in a different way by combining elements in a new pattern or proposing alternative solutions. | Adapt, Build, Change, Choose, Combine, Compile, Compose, Construct, Create, Delete, Design, Develop, Discuss, Elaborate, Estimate, Formulate, Happen, Imagine, Improve, Invent, Make up, Maximize, Minimize, Modify, Original, Originate, Plan, Predict, Propose, Solution, Solve, Suppose, Test, Theory |