Q. P. Code: 23442 Answer Key

Time: 3 hours **Marks: 80**

N. B: 1. Question 1 is compulsory.
2. Attempt any three out of remaining.
3. Assume suitable data if required.

Qu-1 **Attempt any four questions**

a) Consider a suitable relation schema and perform nested query and query using group by clause.

Ans **Consider the following Schema:**
Sailors (sid, sname, rating, age)
Boats (bid, name, color)
Reserves (sid, bid, day)
a) Nested Query:

Select S.sname from Sailors S where S.sid IN (Select R.sid from Reserves R where R.bid=103.
b) Group By clause

Select S.rating, MIN (S.age) AS minage from sailors S where S.age>=18 group by S.rating having count (*)>1

b) Explain ECA Model.

Ans Event condition action (ECA) is a short-cut for referring to the structure of active rules in event driven architecture and active database systems.
Such a rule traditionally consisted of three parts:
The event part specifies the signal that triggers the invocation of the rule
The condition part is a logical test that, if satisfied or evaluates to true, causes the action to be carried out
The action part consists of updates or invocations on the local data
Any example of ECA Model

c) What is view? Discuss the difference between a view and base relation.

Ans A view is a **virtual** or **derived relation**: a relation that does not necessarily exist in its own right, but may be dynamically derived from one or more **base relations**.

Purpose of Views:

The view mechanism is desirable for several reasons:

- It provides a powerful and flexible security mechanism by hiding parts of the database from certain users. Users are not aware of the existence of any attributes or tuples that are missing from the view.
- It permits users to access data in a way that is customized to their needs, so that the same data can be seen by different users in different ways, at the same time.
- It can simplify complex operations on the base relations. For example, if a view is defined as a combination (join) of two relations , users may now perform more simple operations on the view, which will be translated by the DBMS into equivalent operations on the join.

Summary of advantages/disadvantages of views in SQL:

	Advantages
1	Data independence: A view can present a consistent, unchanging picture of the structure of the database, even if the underlying source tables are changed
2	Currency: Changes to any of the base tables in the defining query are immediately reflected in the view.
3	Improved security: Each user can be given the privilege to access the database only through a small set of views that contain the data appropriate for that user, thus restricting and controlling each user's access to the database.
4	Reduced complexity: A view can simplify queries, by drawing data from several tables into a single table, thereby transforming multi-table queries into single-table queries.
5	Convenience: Views can provide greater convenience to users as users are presented with only that part of the database that they need to see. This also reduces the complexity from the user's point of view.
6	Advantages Customization: Views provide a method to customize the appearance of the database, so that the same underlying base tables can be seen by different users in different ways.
7	Data integrity: If the WITH CHECK OPTION clause of the CREATE VIEW statement is used, then SQL ensures that no row that fails to satisfy the WHERE clause of the defining query is ever added to any of the underlying base table(s) through the view, thereby ensuring the integrity of the view.
	Disadvantages
1	**Update restriction: ???**
2	**Structure restriction:** The structure of a view is determined at the time of its creation. If the defining query was of the form SELECT * FROM . . . , then the * refers to the columns of the base table present when the view is created. If columns are subsequently added to the base table, then these columns will not appear in the view, unless the view is dropped and recreated.
3	Performance: There is a performance penalty to be paid when using a view. In some cases, this will be negligible; in other cases, it may be more problematic. For example, a view defined by a complex, multi-table query may take a long time to process as the view resolution must join the tables together *every time the view is accessed*.

d) Define a lock and describe the types of locks used in concurrency control.

Ans A lock is a variable associated with a data item that describes the status of the item with respect to possible operations that can be applied to it. Generally, there is one lock for each data item in the database. Locks are used as a means of synchronizing the access by concurrent transactions to the database items.

1) Binary Lock:

A **binary lock** can have two **states** or **values:** locked and unlocked (or 1 and 0, for

simplicity). A distinct lock is associated with each database item *X*. If the value of the lock on *X* is 1, item *X cannot be accessed* by a database operation that requests the item. If the value of the lock on *X* is 0, the item can be accessed when requested, and the lock value is changed to 1.We refer to the current value (or state) of the lock associated with item *X* as **lock**(***X***).

2) Shared/Exclusive (or Read/Write) Locks.

The preceding binary locking scheme is too restrictive for database items because at most, one transaction can hold a lock on a given item.We should allow several transactions to access the same item X if they all access X for reading purposes only. This is because read operations on the same item by different transactions are not conflicting (see Section 21.4.1). However, if a transaction is to write an item X, it must have exclusive access to X. For this purpose, a different type of lock called a multiple-mode lock is used. In this scheme—called shared/exclusive or read/write locks—there are three locking operations: read_lock(X), write_lock(X), and unlock(X). A lock associated with an item X, LOCK(X), now has three possible states: read-locked, write-locked, or unlocked. A read-locked item is also called share-locked because other transactions are allowed to read the item, whereas a write-locked item is called exclusive-locked because a single transaction exclusively holds the lock on the item.

e) List differentiation between OLTP and OLAP

Ans

	OLTP	OLAP
Source of data	Operational data; OLTPs are the original source of the data.	Consolidation data; OLAP data comes from the various OLTP Databases
Purpose of data	To control and run fundamental business tasks	To help with planning, problem solving, and decision support
What the data	Reveals a snapshot of ongoing business processes	Multi-dimensional views of various kinds of business activities
Inserts and Updates	Short and fast inserts and updates initiated by end users	Periodic long-running batch jobs refresh the data
Queries	Relatively standardized and simple queries Returning relatively few records	Often complex queries involving aggregations
Processing Speed	Typically very fast	Depends on the amount of data involved; batch data refreshes and complex queries may take many hours; query speed can be improved by creating indexes
Space Requirements	Can be relatively small if historical data is archived	Larger due to the existence of aggregation structures and history data; requires more indexes than OLTP
Database Design	Highly normalized with many tables	Typically de-normalized with fewer tables; use of star and/or snowflake schemas
Backup and Recovery	Backup religiously; operational data is critical to run the business, data loss is likely to entail significant monetary loss and legal liability	Instead of regular backups, some environments may consider simply reloading the OLTP data as a recovery method.

Qu-2 a) What is SQLJ used for? Describe the two types of iterators available in SQLJ.

Ans SQLJ: a standard for embedding SQL in Java

- An SQLJ translator converts SQL statements into Java
  – These are executed thru the *JDBC* interface
- Certain classes have to be imported
  – E.g., **java.sql**

Iterator Available in SQLJ

1) Named iterator:
-it is associated with query result by listing the attribute names and
Types that appear in the query result.
2) Positional iterator:
- It list only the attribute types that appear in the query result.

```
//Program Segment J1:
1)   ssn = readEntry("Enter a Social Security Number: ") ;
2)   try {
3)      #sql{select FNAME, MINIT, LNAME, ADDRESS, SALARY
4)         into :fname, :minit, :lname, :address, :salary
5)         from EMPLOYEE where SSN = :ssn} ;
6)   }  catch (SQLException se) {
7)      System.out.println("Social Security Number does not exist: " + ssn) ;
8)      Return ;
9)      }
10)  System.out.println(fname + " " + minit + " " + lname + " " + address + " " +
     salary)
```

b) Differentiate between static and dynamic SQL? Which one is more efficient?

Ans **Static Sql:**

- Most SQL statements can be embedded in a general-purpose *host* programming language such as COBOL, C, Java
- An embedded SQL statement is distinguished from the host language statements by enclosing it between **EXEC SQL** or **EXEC SQL BEGIN** and a matching **END-EXEC** or **EXEC SQL END** (or semicolon)

– Syntax may vary with language

– *Shared variables* (used in both languages) usually prefixed with a colon (:) in SQL

Dynamic SQL

 Objective:

– Composing and executing new (not previously compiled) SQL statements at run-time
 a program accepts SQL statements from the keyboard at run-time

 a point-and-click operation translates to certain SQL query
 Dynamic update is relatively simple; dynamic query can be complex
– because the type and number of retrieved attributes are unknown at compile time

Here is the comparison of static (embedded) SQL and dynamic (interactive) SQL:

SN	Static (embedded) SQL	Dynamic (interactive) SQL
1.	In static SQL how database will be accessed is predetermined in the embedded SQL statement.	In dynamic SQL, how database will be accessed is determined at run time.
2.	It is more swift and efficient.	It is less swift and efficient.
3.	SQL statements are compiled at compile time.	SQL statements are compiled at run time.
4.	Parsing, validation, optimization, and generation of application plan are done at compile time.	Parsing, validation, optimization, and generation of application plan are done at run time.
5.	It is generally used for situations where data is distributed uniformly.	It is generally used for situations where data is distributed non-uniformly.
6.	EXECUTE IMMEDIATE, EXECUTE and PREPARE statements are not used.	EXECUTE IMMEDIATE, EXECUTE and PREPARE statements are used.
7.	It is less flexible.	It is more flexible.

Qu-3 a) Describe ARIES recovery algorithm with example.

Ans The ARIES Recovery Algorithm :

• The Transaction table and the Dirty Page table

– For efficient recovery following tables are also stored in the log during checkpointing:

• Transaction table: Contains an entry for each active transaction, with information such as transaction ID, transaction status and the LSN of the most recent log record for the transaction.

Dirty Page table:

Contains an entry for each dirty page in the buffer, which includes the page ID and the LSN corresponding to the earliest update to that page.

• Checkpointing:

– A check pointing does the following:

• Writes a begin_checkpoint record in the log

• Writes an end_checkpoint record in the log. With this record the contents of transaction table and dirty page table are appended to the end of the log.

• Writes the LSN of the begin_checkpoint record to a special file. This special file is accessed during recovery to locate the last checkpoint information.

– To reduce the cost of checkpointing and allow the system to continue to execute transactions, ARIES uses "fuzzy checkpointing".

• The following steps are performed for recovery

– Analysis phase: Start at the begin_checkpoint record and proceed to the end_checkpoint record. Access transaction table and dirty page table are appended to the end of the log. Note that during this phase some other log records may be written to the log and transaction table may be modified. The analysis phase compiles the set of redo and undo to be performed and ends.

(a)

Lsn	Last_lsn	Tran_id	Type	Page_id	Other_information
1	0	T_1	update	*C*	...
2	0	T_2	update	*B*	...
3	1	T_1	commit		...
4	begin checkpoint				
5	end checkpoint				
6	0	T_3	update	*A*	...
7	2	T_2	update	*C*	...
8	7	T_2	commit		...

(b)

TRANSACTION TABLE

Transaction_id	Last_lsn	Status
T_1	3	commit
T_2	2	in progress

DIRTY PAGE TABLE

Page_id	Lsn
C	1
B	2

(c)

TRANSACTION TABLE

Transaction_id	Last_lsn	Status
T_1	3	commit
T_2	8	commit
T_3	6	in progress

DIRTY PAGE TABLE

Page_id	Lsn
C	1
B	2
A	6

Figure 19.6
An example of recovery in ARIES. (a) The log at point of crash. (b) The Transaction and Dirty Page Tables at time of checkpoint. (c) The Transaction and Dirty Page Tables after the analysis phase.

b) Explain Indexing Technique in the database.

Ans

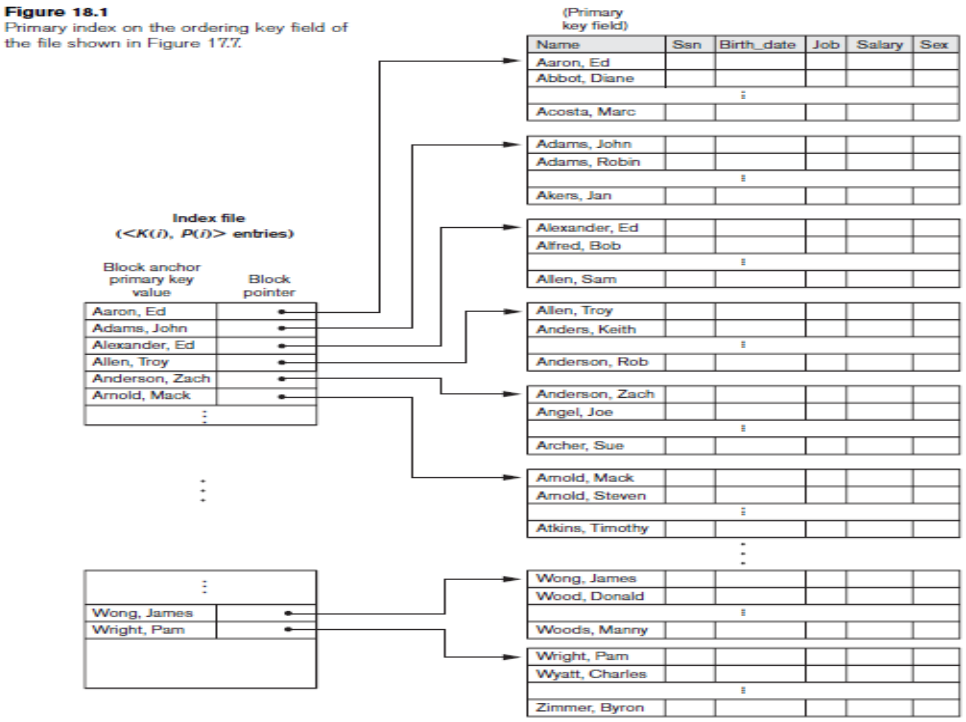
- Indexing is a data structure technique to efficiently retrieve records from database files based on some attributes on which the indexing has been done. Indexing in database systems is similar to the one we see in books. Indexing is defined based on its indexing attributes. Indexing can be one of the following types:
- Primary Index: If index is built on ordering 'key-field' of file it is called Primary Index. Generally it is the primary key of the relation.
- Secondary Index: If index is built on non-ordering field of file it is called Secondary Index.
- Clustering Index: If index is built on ordering non-key field of file it is called Clustering Index. Ordering field is the field on which the records of file are ordered. It can be different from primary or candidate key of a file. Ordered Indexing is of two types:
- Dense Index

□□Sparse Index

In dense index, there is an index record for every search key value in the database. This makes searching faster but requires more space to store index records itself. Index record contains search key value and a pointer to the actual.

Multilevel Index:

Index records are comprised of search-key value and data pointers. This index itself is stored on the disk along with the actual database files. As the size of database grows so does the size of indices. There is an immense need to keep the index records in the main memory so that the search can speed up. If single level index is used then a large size index cannot be kept in memory as whole and this leads to multiple disk accesses.



Qu-4 a) Find the cost of data transfer over the network for following details.
Employee table is at site 1with 10,000 rows. Each row size is 100 bytes.
Department table is at site 2 with 100 rows. Each row size is 35 bytes.
Find optimum solution for data transfer if following query is executed from site 3.
Query: For each employee retrieve the emp_name and dept_name where employee works.
Size of result tuple is 40 bytes.

Ans Employee at site 1 and Department at Site 2

- Employee at site 1. 10,000 rows. Row size = 100 bytes. Table size = 106 bytes.
- Department at Site 2. 100 rows. Row size = 35 bytes. Table size = 3,500 bytes.

The result of this query will have 10,000 tuples, assuming that every employee is related to a department.
Suppose each result tuple is 40 bytes long. The query is submitted at site 3 and the result is sent to this site.

- Problem: Employee and Department relations are not present at site 3.
- Strategies:
  Transfer Employee and Department to site 3.
  Total transfer bytes = 1,000,000 + 3500 = 10,03,500 bytes.
  Transfer Employee to site 2, execute join at site 2 and send the result to site 3.
  Query result size = 40 * 10,000 = 400,000 bytes.
  Total transfer size = 400,000 + 1,000,000 = 14,00,000 bytes.
  Transfer Department relation to site 1, execute the join at site 1, and send the result to site 3.
  Total bytes transferred = 4,00,000 + 3500 = 4,03,500 bytes.

 Optimization criteria: minimizing data transfer so strategy 3.

b) Explain different ways of concurrency control in DDBMS

Ans

- Distributed Databases encounter a number of concurrency control and recovery problems which are not present in centralized databases. Some of them are listed below.
  - Dealing with multiple copies of data items
  - Failure of individual sites
  - Communication link failure
  - Distributed commit
  - Distributed deadlock
  - Dealing with multiple copies of data items:
    - The concurrency control must maintain global consistency. Likewise the recovery mechanism must recover all copies and maintain consistency after recovery.
  - Failure of individual sites:
    Database availability must not be affected due to the failure of one or two sites and the recovery scheme must recover them before they are available for use.
  - Communication link failure:
    - This failure may create network partition which would affect database availability even though all database sites may be running.
  - Distributed commit:
    - A transaction may be fragmented and they may be executed by a number of sites. This require a two or three-phase commit approach for transaction commit.
    - Since transactions are processed at multiple sites, two or more sites may get involved in deadlock. This must be resolved in a distributed manner.
  - Distributed Concurrency control based on a distributed copy of a data item
  - Primary Copy Technique:
  - Primary site approach with no backup site:

Concurrency control based on voting:

- There is no primary copy of coordinator.
- Send lock request to sites that have data item.
- If majority of sites grant lock then the requesting transaction gets the data item.
- Locking information (grant or denied) is sent to all these sites.

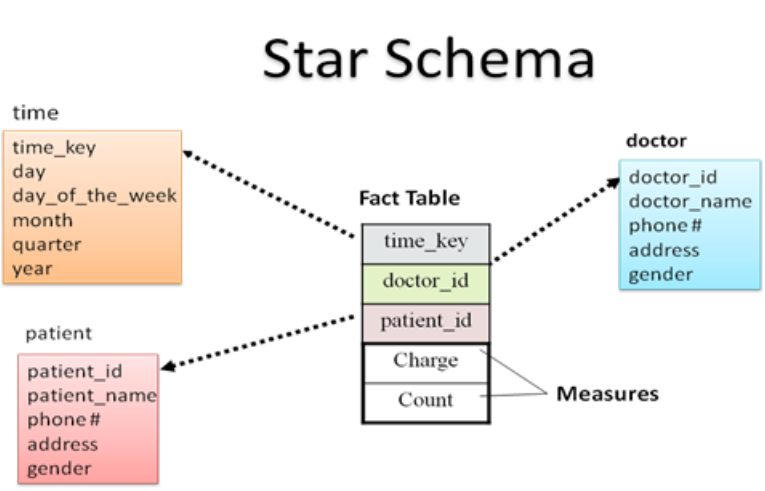
Qu-5 Consider a data ware house for a hospital where there are three dimensions:
1) Doctor 2) Patient and 3) Time

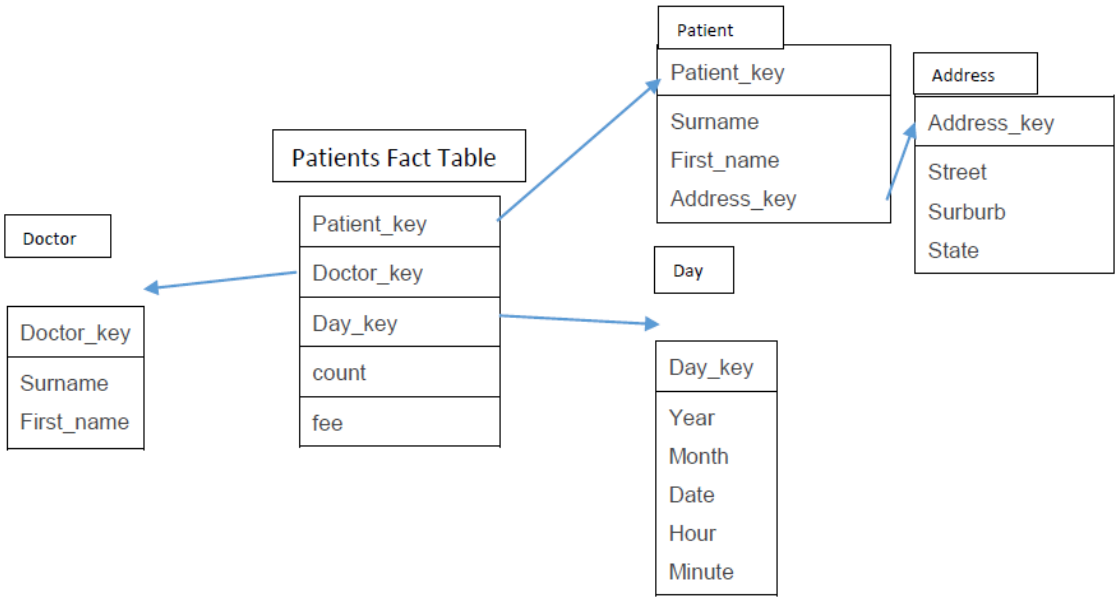
And two measures count and charge.

Using above example perform following

i) STAR schema
ii) Snowflake schema
iii) Rollup & Drilldown operations
iv) Pivot operation
v) Slice and Dice operations

Ans





The operations to be performed are:
Roll-up on *time* from *day* to *year* or Roll-up on *patient* from individual patient to all, the drilldown operation is reverse of rollup.
Slice for *time=Some value*.

Qu-6 Explain the following concepts with the help of examples.

a) SQL Injection

Ans **SQL Injection Methods**

- **SQL Manipulation.** A manipulation attack, which is the most common type of injection attack, changes an SQL command in the application
  SELECT * FROM users **WHERE** username = 'jake' and PASSWORD = 'jakespasswd'.

  The attacker can try to change (or manipulate) the SQL statement, by changing it as follows:

  SELECT * FROM users **WHERE** username = 'jake' and (PASSWORD = 'jakespasswd' or 'x' = 'x')

  Code Injection:
  - This type of attack attempts to add additional SQL statements or commands to the existing SQL statement by exploiting a computer bug, which is caused by

processing invalid data. The attacker can inject or introduce code into a computer program to change the course of execution. Code injection is a popular technique for system hacking or cracking to gain information.

- **SELECT * FROM users WHERE name = 'a';DROP TABLE users; SELECT * FROM userinfo WHERE 't' = 't';**

Function Call Injection.

- In this kind of attack, a database function or operating system function call is inserted into a vulnerable SQL statement to manipulate the data or make a privileged system call.
- For example, it is possible to exploit a function that performs some aspect related to network communication. In addition, functions that are contained in a customized database package, or any custom database function, can be executed as part of an SQL query

b) Mandatory Access Control

Ans The discretionary access control technique of granting and revoking privileges on relations has traditionally been the main security mechanism for relational database systems. This is an all-or-nothing method: A user either has or does not have a certain privilege. In many applications, an additional security policy is needed that classifies data and users based on security classes. This approach, known as mandatory access control, would typically be combined with the discretionary access control mechanisms.

Typical security classes are top secret (TS), secret (S), confidential (C), and unclassified (U), where TS is the highest level and U the lowest. Other more complex security classification schemes exist, in which the security classes are organized in a lattice. For simplicity, we will use the system with four security classification levels, where TS $\geq$ S $\geq$ C $\geq$ U, to illustrate our discussion. The commonly used model for multilevel security, known as the Bell-LaPadula model, classifies each subject (user, account, program) and object (relation, tuple, column, view, operation) into one of the security classifications TS, S, C, or U. We will refer to the clearance (classification) of a subject S as class(S) and to the classification of an object 0 as class(D). Two restrictions are enforced on data access based on the subject/object classifications:

1. A subject S is not allowed read access to an object 0 unless class(S) ~ class(O). This is known as the simple security property.
2. A subject S is not allowed to write an object 0 unless class(S) ~ class(O). This is known as the star property (or *-property).

c) Statistical Database

Ans Statistical databases are used mainly to produce statistics on various populations.
The database may contain confidential data on individuals, which should be protected from user access. Users are permitted to retrieve statistical information on the populations, such as averages, sums, counts, maximums, minimums, and standard deviations. A population is a set of tuples of a relation (table) that satisfy some selection condition. Statistical queries involve applying statistical functions to a population of tuples. For example, we may want to retrieve the *number* of individuals in a population or the *average income* in the population.
However, statistical users are not allowed to retrieve individual data, such as the income of a specific person. Statistical database security techniques must prohibit the retrieval of

individual data. This can be achieved by prohibiting queries that retrieve attribute values and by allowing only queries that involve statistical aggregate functions such as COUNT, SUM, MIN, MAX, AVERAGE, and STANDARD DEVIATION.
Such queries are sometimes called statistical queries.

d) Timestamp Ordering Protocol

Ans **Timestamp Ordering (TO).** Whenever some transaction T tries to issue a read_item(X) or a write_item(X) operation, the basic TO algorithm compares the timestamp of T with read_TS(X) and write_TS(X) to ensure that the timestamp order of transaction execution is not violated. If this order is violated, then transaction T is aborted and resubmitted to the system as a new transaction with a new timestamp. If T is aborted and rolled back, any transaction T1 that may have used a value written by T must also be rolled back. Similarly, any transaction T2 that may have used a value written by T1 must also be rolled back, and so on. This effect is known as cascading rollback and is one of the problems associated with basic TO, since the schedules produced are not guaranteed to be recoverable. An additional protocol must be enforced to ensure that the schedules are recoverable, cascadeless, or strict. We first describe the basic TO algorithm here. The concurrency control algorithm must check whether conflicting operations violate the timestamp ordering in the following two cases:

1. Whenever a transaction T issues a write_item(X) operation, the following is checked:
    a. If read_TS(X) > TS(T) or if write_TS(X) > TS(T), then abort and roll back T and reject the operation. This should be done because some younger transaction with a timestamp greater than TS(T)—and hence after T in the timestamp ordering—has already read or written the value of item X before T had a chance to write X, thus violating the timestamp ordering.
    b. If the condition in part (a) does not occur, then execute the write_item(X) operation of T and set write_TS(X) to TS(T).
2. Whenever a transaction T issues a read_item(X) operation, the following is checked:
    a. If write_TS(X) > TS(T), then abort and roll back T and reject the operation. This should be done because some younger transaction with timestamp greater than TS(T)—and hence after T in the timestamp ordering—has already written the value of item X before T had a chance to read X.
    b. If write_TS(X) ≤ TS(T), then execute the read_item(X) operation of T and set read_TS(X) to the larger of TS(T) and the current read_TS(X).