BLOCK II: DATABASE QUERY, TRANSACTION PROCESSING AND SECURITY CONCEPTS

Unit 1 : Query Processing and Optimization
Unit 2 : Transaction Processing
Unit 3 : Concurrency Control and Recovery Techniques
Unit 4 : Database Security

UNIT 1: QUERY PROCESSING AND OPTIMIZATION

Unit Structure:

1.1 Introduction

1.2 Objectives

1.3 Query Processing

1.3.1 Block Diagram to show the steps of conversion

1.3.2 Equivalence Rules

1.4 Query Optimization

1.4.1 Query Cost Measurement

1.4.2 Generation of Equivalence Expression

1.4.3 Transformation Examples

1.4.4 Choosing the Evaluation Plans

1.4.5 Estimation for the Statistics of Costs

1.4.6 Maintaining the materialized View

1.5 Summing Up

1.6 Answers to Check Your Progress

1.7. Possible Questions

1.8 References and Suggested Readings

Space for learners:

1.1 INTRODUCTION

Space for learners:

In this unit, you will get to learn in detail about query processing and query optimization and how their implementations are done in Database Management System (DBMS). You already know that databases are created to organize and store all the related data and information at a particular place so that the users can access and manipulate it as per requirement. It may so happen that different users may use their languages to access those data, which at times becomes difficult for the DBMS system to return the actual information, and also the data may not be accurate. So there comes the need for a common operation or you can say a language that can communicate between the system and the user.

1.2 UNIT OBJECTIVES

Studying this unit, you will be able to:

- Understand the concept of query processing
- Understand the concept of parsing, translation
- Understand the concept of optimization
- Know how to choose the evaluation plan
- learn about different execution plans which are cost-effective and generate a code optimizer

1.3 QUERY PROCESSING

To bridge the gap between the DBMS and the Users, a standard language, which is understandable by the DBMS is used between the two, so that correct data are retrieved on processing. This standard language is known as the Structured Query Language (SQL) which is a High-Level Language (HLL). DBMS automatically converts this HLL to Low-Level language (LLL) which is a machine-understandable language using Relational Algebra whenever a query is encountered by the system [1].

A query is submitted to the database to retrieve relevant information from it. Whenever any Query is encountered by the system, verification of the query is done by the DBMS and that query is converted to some Low-Level Language after which path of the execution is selected and the queried data gets retrieved from the storage memory. It is done internally by the DBMS. This whole technique is performed by the Query Processor which is a feature present in the DBMS and the functions that are performed are termed as the Query Processing.

As discussed, DBMS consists of a Query Processor that checks and verifies the queries given by the Users which are in the form of SQL commands (HLL). It then translates these commands into Low-Level Language that is understandable to the DBMS system so that the system can work on it. It converts the HLL Step-by-Step into LLL and returns the value which the users want. In query processing, sorting is a very important step. For performing the operations like ORDER-BY, JOIN, PROJECT, SELECT, UNION, INTERSECTION, etc. sorting, or merge-sort algorithm is a very important phase. A term called 'external sorting' is applied when the memory is small and records of large files need to be stored. It consists of two phases – the sorting phase and the merging phase.

In the case of sorting phase, the files are sorted very precisely so that it easily fits into the available memory. The number of executions is specified to its initial and dedicated file blocks and also on the amount of buffer space.

In the case of the merging phase, the sorted executed query is merged at different passes. The amount of merge that can be occurred together is termed the degree of a merge. For each pass at least one buffer block is required for holding the executed query.

The processing of a query is performed in the following given steps.

a) Parsing and Translation of the Query.
b) Optimization of the parsed query.
c) Compilation or Interpretation in the Query Code Generator.
d) Execution into the Evaluation engine.

Space for learners:

1.3.1 The Block Diagram to Show the Steps of Conversion in Query Processing

Space for learners:

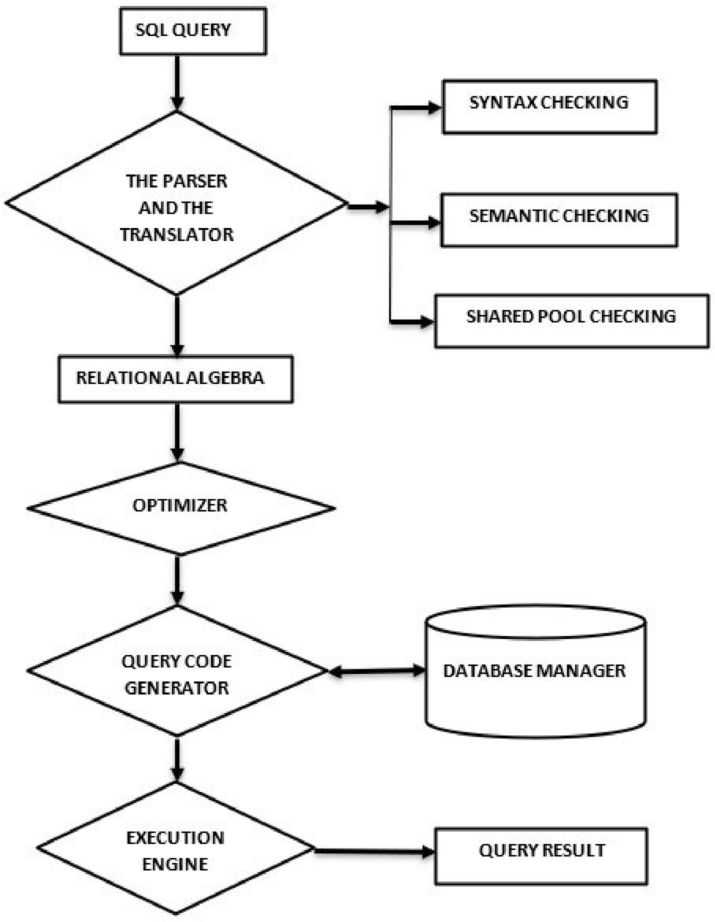


Figure 1: Block Diagram for Query Processing

STEP 1.PARSING –

An SQL query is communicated to the DBMS system through an application program. The high-level language is translated into its low-level language. That is, the SQL query is converted into its relational algebra.

Query blocks are formed after the division of the original SQL query, and hence it becomes ready for its optimization. Query blocks can contain a single expression of SELECT, WHERE, FROM, HAVING, GROUP BY. When the nested queries are encountered having aggregate operators like COUNT, SUM, MIN, MAX, then a separate query block is formed.

For example, the following query is encountered by the system where the teacher wants to fetch the name of all the students where age must be less than 20 years.

select stud_name from Student where age<20;

This query is converted into its relational algebra as given below.

$$\pi_{age} (\sigma_{age} < 20 \text{ (Student)})$$

On converting the query into its relational algebra internally, other steps of parsing are done. When this query is encountered with the system, scanning of the same is done to check whether any syntax error is present in the query or not.

- After proper validation, this relational algebra is then converted into tokenized form (from the example above 'select', 'stud_name', 'from', 'student', 'where', 'age<20' are different tokens) represented as a Parse tree.
- After that, the Parser performs some checks such as Syntax, Semantic, and Shared Pool checks as shown in the diagram. In the Syntax checking the parser checks whether the syntax of the query is correct or not. An example is shown below.

select stud_name form Student through age<20;

It can be seen that '**form**' and '**through**' are the words respectively spelled incorrectly and wrongly written. These are the syntax errors.

- The semantic check helps in checking whether the table, keywords, columns present in the query are also present in the Database or not. If it is present, then it proceeds to the next step and if it is not present then it returns an error to the User.
- During its execution, every query is given a hash code by the parser and if this code is present then no extra performances are done in the rest of the steps. This hash code is checked by the Shared Pool [2].

Space for learners:

Space for learners:

STEP 2.OPTIMIZATION –

After the query goes through different processes in the Parsing phase, the parsed query is then shifted to the Optimization phase so that a minimal cost of execution is selected. Minimal evaluation cost is the time when a query is encountered in the system till it returns the result. For selecting the best minimal plan for executing the query, a Catalog Manager, present in the Optimizer helps in the selection of the minimal cost. A hard parsing must be performed for at least a single DML statement and after that, the process of optimization is carried on. Access routines help in the implementation of query operations such as SELECT, JOIN, PROJECT, etc.

Typically, optimization takes any one of the following forms: The heuristic form of optimization or Cost based form of optimization.

(a) Heuristic optimization – With the help of heuristic rules, refinement of query execution is done so that individual operations are reordered.

(b) Cost-based optimization – with the help of cost-based rules, estimation of the cost of plans is done by the reduction of the overall cost of the query.

STEP 3. QUERY CODE GENERATION –

Executions plans are none other than the systematic way of ordering the access routines. Once the execution plans are selected and determined by the optimization process, it is the work of the code generator to determine the actual access routines needed to be executed. The query code is then compiled or interpreted and it is then transferred to the database. The database processor then helps the query for its execution. The execution plans are stored in the database.

STEP 4. EXECUTION –

After the query has gone through the processes of Parsing, Optimization, and Query Code Generation, it is passed into the execution phase. The results of the query are executed and then it is returned to the Users along with its runtime errors.

1.3.2 Equivalence Rules

Space for learners:

Following are the equivalence rules used during the time of processing and optimizations.

1)SIGMA-CASCADE - Intersection of $\boldsymbol{\theta}1$ and $\boldsymbol{\theta}2$ makes the system very much expensive therefore inner selection of $\boldsymbol{\theta}2$ and outer selection of $\boldsymbol{\theta}1$ is done to make the system effective.

$$\sigma_{\boldsymbol{\theta}1 \wedge \boldsymbol{\theta}2}(\mathrm{E}) = \sigma_{\boldsymbol{\theta}1}(\sigma_{\boldsymbol{\theta}2}(\mathrm{E}))$$

2)COMMUTATIVE SELECTION – Since σ is commutative therefore the practical implementation of it is necessary.

$$\sigma_{\boldsymbol{\theta}1}(\sigma_{\boldsymbol{\theta}2}(\mathrm{E})) = \sigma_{\boldsymbol{\theta}2}(\sigma_{\boldsymbol{\theta}1}(\mathrm{E}))$$

3)PI- CASCADE – Combining all the projections into a single projection is a very good option.

$$\pi_{L1}(\pi_{L2}(\ldots(\pi_{Ln}(E)).) = \pi_{L1}(E)$$

4)CARTESIAN PRODUCTS AS THETA-JOINS –

(a) EQUIVALENCE 1 – Only using the cross product makes the system very expensive, so a theta join is also used to make it effective.

$$\sigma_{\boldsymbol{\theta}}(E_1 \times E_2) = E_{1\boldsymbol{\theta}} \bowtie E_2$$

(b)EQUIVALENCE 2 – If both the thetas are joined, then it will require little execution support.

$$\sigma_{\boldsymbol{\theta}1}(E_1 \bowtie_{\boldsymbol{\theta}2} E_2) = E_1 \bowtie_{\boldsymbol{\theta}1 \wedge \boldsymbol{\theta}2} E_2$$

5) THETA JOINS ARE COMMUTATIVE – Since theta joins are commutative, therefore the query processing depends upon the inner and outer joins.

$$E_1 \bowtie_{\theta} E_2 = E_2 \bowtie_{\theta} E_1$$

Space for learners:

6)NATURAL JOIN – Since joins are both commutative and associative, therefore the tables having lesser entries must be joined.

$$(E_1 \bowtie E_2) \bowtie E_3 = E_1 \bowtie (E_2 \bowtie E_3)$$

7) DISTRIBUTION OF SELECTION OPERATION –

(a) EQUIVALENCE 1 – Theta join is performed, after selection is applied. Then E_1 is joined with E_2.

$$\sigma_{\theta 1 \wedge \theta 2} (E_1 \bowtie_{\theta} E_2) = (\sigma_{\theta 1}(E_1)) \bowtie (\sigma_{\theta 2}(E_2))$$

(b) EQUIVALENCE 2 – Here $\theta 1$ and $\theta 2$ contains the attribute of E_1 and E_2 respectively.

$$\sigma_{\theta 0} (E_1 \bowtie_{\theta} E_2) = (\sigma_{\theta 0} (E_1) \bowtie_{\theta} E_2$$

8) THETA-JOIN PROJECTION –

(a) EQUIVALENCE 1 – Here L_1 is projected over E_1 and L_2 is projected over E_2. It is compulsory for doing projections before joining.

$$\pi_{L1 \cup L2} (E_1 \bowtie_{\theta} E_2) = (\pi_{L1}(E_1)) \bowtie_{\theta} (\pi_{L1}(E_2))$$

(b) EQUIVALENCE 2 - Here L_1 is projected over E_1 and L_2 is projected over E_2. It is compulsory for doing projections before joining. E_3 is an equivalence relation that joins both the relation using L_3.

$$\pi_{L1 \cup L2} (E_1 \bowtie_{\theta} E_2) = \pi_{L1 \cup L2} ((\pi_{L1 \cup L3})) \bowtie_{\theta} (\pi_{L2 \cup L4} (E_2))$$

Space for learners:

9) UNION AND INTERSECTION ARE COMMUTATIVE

$$E1 \cup E2 = E2 \cup E1$$

$$E1 \cap E2 = E2 \cap E1$$

10) UNION AND INTERSECTION ARE ASSOCIATIVE

$$(E_1 \cup E_2) \cup E_3 = E_1 \cup (E_2 \cup E_3)$$

$$(E_1 \cap E_2) \cap E_3 = E_1 \cap (E_2 \cap E_3)$$

11) SELECTION OPERATION OVER DIFFERENCE

$$\sigma_P(E_1 - E_2) = \sigma_P(E_1) - \sigma_P(E_2)$$

12) PROJECTION OVER UNION

$$\pi_L (E_1 \cup E_2) = (\pi_L (E_1)) \cup (\pi_L (E_2))$$

1.4 QUERY OPTIMIZATION

As discussed in the previous section, query optimization is performed by the DBMS for selecting the minimal cost of query execution, that is the time when a query is encountered in the system till it returns the result. It is an automated process. The optimizer selects for the query an efficient plan for its execution. This selection of the plan is performed by the catalog manager. Most of the structures of the query optimizers can be seen as the Left-deep Join orders. For pushing the selections and the projections through the query tree, a heuristic approach is used. This heuristic approach chose the best relation for joining the next relations. This left deep Join tree also helps in reducing the complexities caused by optimizations. Every input made on the right-hand side of the tree is a relation and not a join. In the figure given below, R1, R2, R3, R4 are the relations.

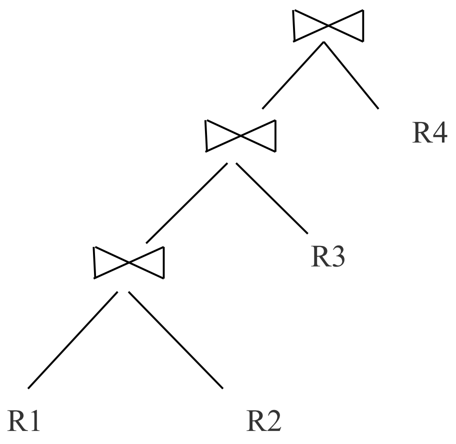


Figure 2: LEFT - DEEP JOIN ORDER

1.4.1 Query Cost Measurement

Different DBMS structures balance the query plan and the choice of execution quality in many different possible ways. For these query plans and the choice of execution quality, the evaluation of the best Cost-based optimization is performed by the optimizer by choosing a minimal cost.There may be different types of cost efficiencies depending upon the need and situations, which may include

a) Minimization of the processing time
b) Minimization of the response time
c) Minimization of the input or output time
d) Minimization of the network time
e) Combination of the above situations etc.

Internally it can be seen that retrieving the desired data by the query, both from the primary and the secondary memories relatively takes an ample amount of time. This stipulated time is taken by different factors such as CPU time, disk I/O time, network access time, etc. It is seen that Disk I/O (finding the records by the processor from the secondary memory and returning the result) is the most efficient of all but it takes a much larger amount of time than the others. Mainly two factors are considered for the calculation of the Disk I/O access time. They are the

Seek Time and the Transfer time. Let us understand this with the help of an example.

Suppose you need to find the **AGE** of the student whose name is '**Peter**' from the STUDENT table. So, the Disk I/O would work on the factors Seek and Transfer time as follows;

a) SEEK TIME – The amount of time required in finding a single record by the Processor in the disk memory is termed as the seek time. Seek time is usually represented by ***tS***. For the example given above, the time, from accessing the disk block in the memory till the search of the AGE of the student Peter is the Seek time of the Disk[1].

b) TRANSFER TIME –The amount of time required by the disk in returning the query result to the user or the processor is termed the transfer time. Transfer time is usually represented as ***tT***. For the example given above, the time of returning the value of the age of Peter to the User or the processor is known as the Transfer time of the Disk[1].

1.4.2 Generation of Equivalence Expression

You are already familiar with the fact that relational algebras have equivalent expressions if they generate tuples of the same form, no matter what is the order of their tuples. In SQL statements, input expression and output expression have tuples having multiset values and replace each other's form.

1.4.3 Transformation Examples

- **PUSHING SELECTIONS**[3]

Pushing selection helps in the reduction of the Relation's size if it is performed in an early stage. An example of a query is given as "Fetch the Names of all the Employees in an IT department, along with the job profile of the Posts in which they work".

$$\prod_{name,\ post}(\sigma_{dept\_name\ =\ \text{“IT”}}(\text{employees} \bowtie (\text{works} \bowtie \prod_{job\_profile,\ post}(\text{profile}))))$$

Space for learners:

By transforming the above expression with the help of rule 7 (a) of the Equivalence Rule you will get the following expression

$$\prod_{name,\ post}((\sigma_{dept\_name\ =\ \text{“IT”}}(\text{employees})) \bowtie (\text{works} \bowtie \prod_{job\_profile,\ post}(\text{profile})))$$

• **MULTIPLE TRANSFORMATION**[3]

An example of a query is given as “Fetch the Names of all the Employees in an IT department for the year 2020, along with the job profile of the Posts in which they worked”.

$$\prod_{name,\ post}(\sigma_{dept\_name\ =\ \text{“IT”}\ \wedge\ year\ =\ 2020}(\text{employees} \bowtie (\text{works} \bowtie \prod_{job\_profile,\ post}(\text{profile}))))$$

By transforming the above expression with the help of rule 6 (a) of the Equivalence Rule you will get the following expression

$$\prod_{name,\ post}(\sigma_{dept\_name\ =\ \text{“IT”}\wedge\ year\ =\ 2020}((\text{employees} \bowtie \text{works}) \bowtie \prod_{job\_profile,\ post}(\text{profile})))$$

• **PUSHING PROJECTIONS**[3]

An example of a query is given as “Fetch the Names of all the Employees in an IT department, along with the job profile of the Posts in which they work”.

$$\prod_{name,\ post}(\sigma_{dept\_name\ =\ \text{“IT”}}(\text{employees}) \bowtie \text{works}) \bowtie \prod_{job\_profile,\ post}(\text{profile}))))$$

By transforming the above expression with the help of rules 8 (a) and 8 (b) of the Equivalence Rule you will get the following expression

$$\prod_{name,\ post}(\prod_{name,\ job\_profile}(\sigma_{dept\_name\ =\ \text{“IT”}}(\text{employees}) \bowtie \text{works}))$$

Space for learners:

$\bowtie \prod_{job\_profile, post}$(profile))))

- **ORDERING OF JOIN**[3]

Join ordering minimizes the storage cost of the relation. For example, there are two conditions of joining attributes – one of them is large and the other is small. Suppose there are three relations R1, R2, R3.

R1 and R2 – Large join (R1 $\bowtie$ R2),

R1 and R3 – Smaller join (R1 $\bowtie$ R3)

If reordering of the Joins are done, you will get

$$(R1 \bowtie R3) \bowtie R2$$

1.4.4 Choosing the Evaluations Plans

You should remember that only choosing the evaluation plan with its cheapest form will not always produce a good result. For solving that problem optimizers have in their systems implemented features such as Heuristic optimization and Cost - based optimizations.

Heuristic Optimization

This type of optimization is very cost-effective and also choices of execution get reduced in this type of system. By using some rules as given below, query tree transformation is done by the heuristic optimizer for improving the performance of its execution.

- Selections must be performed early so that tuples get reduced.
- Projections must be performed early so that attributes get reduced.
- Join operations and restrictive selections are performed.
- Partial cost-based and only heuristic optimizations are used by different users.

Space for learners:

Cost-Based Optimization

Space for learners:

This type of optimization is best for a bigger dataset but is very expensive. The process of conversion of a logical query into its physical query is performed by some rules known as the Equivalence rules. These rules determine what type of algorithms should be applied to the operations for determining the minimal cost of the operations.

Depending upon the equivalence rules, the cost-based optimizer is dependent on the following criteria.

- There must be some efficient techniques for deriving duplicate expressions.
- To avoid the formation of the copies of multiple sub-expressions, the representation of the expressions must be space-efficient.
- At the time of its first optimization, dynamic programming helps in storing the sub-expression based on Memorization and can reuse the sub-expression again and again.
- For avoiding all the plans to get generated, pruning techniques based on cost-based are applied.

1.4.5 Estimation forthe Statisticsof Costs

The cost is estimated as shown below.

1.4.5.1 Cost Estimation

Given below are some of the considerations for cost estimation.

- n_r – for a given relation ‘r’, ‘n_r’ is the number of tuples.
- b_r – for a given relation ‘r’, ‘b_r’ is the number of blocks.
- l_r - for a given relation ‘r’, ‘l_r’ is the size of the tuple.
- f_r - for a given relation ‘r’, ‘f_r’ is the tuples fitting into one block.
- V (A, r) - for a given relation 'r', V (A, r) is the distinct values for an attribute in relation r.
- If the above tuples are put inside a file, then you will have, $b_r = \lceil n_r / f_r \rceil$

1.4.5.2 Size Estimation

- **$\sigma_{A=v}$ (r)**: For estimated size = 1, n_r / V (A, r) is the number of conditions that would satisfy the equality conditions.
- **$\sigma_{A \leq V}$ (r):** For the information having statistical values, it is assumed that the value of **c =** n_r **/2**, where 'c' estimates the number of tuples.It satisfies the following minimum and maximum conditions.
  - If v < min (A, r), then c = 0;
  - C = n_r. (v−min (A, r)) / (max (A, r) −min (A, r))

1.4.6 Maintaining the Materialized View

Computing the contents of the costs and storing it is usually referred to as the materialized view and keeping them up-to-date is termed as materialized view maintenance. Maintenance can also be done by re-computing it. Incremental view maintenance helps in maintaining the relational database changes and updating it. Maintenance of the materialized view can be performed in the following ways:

- For each relation, insert, delete and update triggers are defined manually.
- For updating the database relations, views are updated by code which is manually written.
- Directly linked with the database.

1.4.6.1 Operations Over the Materialized View

1) JOIN OPERATION

Let there be a relation 'r' where two states are present, r^o and r^n**.** Let another relation 's' be also present simultaneously. When we insert any value to the relation 'r' (i^r), we get the following

($r^n \bowtie$ s) and can be rewritten as ($r^o \cup i^r \bowtie$ s) or ($r^o \bowtie$ s)∪ ($i^r \bowtie$ s).

2) SELECTION OPERATION

Space for learners:

Let there be a relation ‘r’ where two views are present, v^o and v^nagainst a single viewv = $\sigma_\theta(r)$. Then the selection of the view on relation r is depicted as ($v^n = v^o \cup \sigma_\theta(i^r)$).

Space for learners:

3) PROJECTION OPERATION

This operation is complicated than the other two operations. It has a single projected tuple termed $\prod_A(r)$.

4) COUNT

Counting the number of tuples is helped by this example ($v = A^g count(B)^{(r)}$).If tuples are already present in the view (v), only the count value is incremented, and if a new tuple needs to be added or subtracted, then the count value is given to be 1.

5) SUM

The Sum of the tuples is found out with this example ($v = A^g sum(B)^{(r)}$).The concept of the sum is almost the same as that of the COUNT operation but instead of updating the count value for addition or subtraction of the tuples, it is needed to update the B value but the order of the count is maintained for each transaction.

CHECK YOUR PROGRESS

Fill in the following blanks:

1. _________helps in identifying the tokens.

2. In query processing, __________ is a very important step.

3. The amount of merge that can be occurred together is termed as the _____________.

4. Query___________ are formed after the division of the original SQL query.

5. The _____________ check helps in checking whether the table, keywords, columns present in the query are also present in the Database or not.

6. With the help of heuristic rules, refinement of query execution is done so that individual operations are ____________.

Space for learners:

8. Most of the structures of the query optimizers can be seen as the ____________ orders.

9. Join ordering ____________ the storage cost of the relation.

10. The amount of time required by the disk in returning the query result to the user or the processor is termed as the __________________.

1.5 SUMMING UP

- Databases are created to organize and store all the related data and information at a particular place so that the users can access and manipulate it as per requirement.
- To bridge the gap between the DBMS and the Users, a standard language known as the SQL, which is understandable by the DBMS is used between the DBMS and the Users, so that correct data are retrieved on processing.
- External Sorting is applied when the memory is small and records of large files need to be stored.
- Query blocks are formed after the division of the original SQL query, and it becomes ready for its optimization.
- When Nested queries are encountered having aggregate operators like COUNT, SUM, MIN, MAX, then a separate query block is formed.
- Executions plans are none other than the systematic way of ordering the Access Routines.
- The selection of the execution plan is done by the Catalog Manager.
- Most of the structures of the query optimizers can be seen as the Left-deep Join orders which help in reducing the complexities and optimizations.

1.6 ANSWERS TO CHECK YOUR PROGRESS

Space for learners:

1. Scanner,
2. Sorting,
3. Degree of merge,
4. Blocks,
5. Semantic,
6. Reordered,
7. Hard parsing,
8. Left Deep join,
9. Minimizes,
10. Transfer time

1.7POSSIBLE QUESTIONS

Short Answer type Questions:

1) Define the terms:

   (a) Scanner, (b) Parser, (c) Parse tree

2) What is query processing?
3) What are the steps of query processing?
4) What happens during the execution phase?
5) Define heuristic optimization.
6) Define cost-based optimization.
7) Define seek time.
8) Define transfer time.
9) Define in brief the pushing projection.

Space for learners:

10) How are the joins ordered?

11) What are some of the considerations for cost optimization?

12) How size is estimated for costs?

Long Answer type Questions:

1) Explain what is Query Processing.

2) Draw the Block Diagram of Query Processing and explain it.

3) Write the Equivalence rules for Query Processing.

4) Explain how Query Cost is measured.

5) Explain transformation with examples.

6) Explain Heuristic and Cost based optimizations.

7) What is the meaning of a materialized view? Explain the operations of the materialized view.

1.8REFERENCES AND SUGGESTED READINGS

[1] https://www.tutorialcup.com/dbms/query-processing.htm

[2] https://www.geeksforgeeks.org/sql-query-processing/

[3] http://www.cbcb.umd.edu/confcour/Spring2014/CMSC424/query_optimization.pdf

[4] Database Systems Models, Languages, Design, and Application Programming by Ramez Elmasri and Shamkant B. Navathe, 5th edition by Pearson

UNIT 2: TRANSACTION PROCESSING

Space for learners:

Unit Structure:

2.1 Introduction

2.2 Learning Objectives

2.3 Transaction processing- transaction and system concepts,

2.4 Desirable properties

2.5 Schedules

2.6 Recoverability

2.6.1 Recoverable Schedule

2.6.2 Cascade less Schedule

2.7 Summing Up

2.8 Answers to Check Your Progress

2,9 Possible Questions

2.10 References and Suggested Readings

2.1 INTRODUCTION

Consider the situation of a database user who is holding two accounts in a bank and wants to transfer some amount from his first account to the second account. The situation looks simple to implement however, it will involve several operations such as reading the balance amount available in the first account and if the balance is sufficient subtract the amount that needs to be transferred followed by updating the final balance in the first account. Then the balance in the second account is checked and the amount that needs to be transferred is added to the balance and the final balance is updated in the account. The operations involved in this example are reading, subtracting, updating, adding and writing. The set of all these operations is grouped into a single unit and is called as a transaction. So, a transaction processing system must ensure that all the operations in the transaction are executed in a proper order and either all operations are executed or none of them are executed. That is, in case of any failure, it should not be that the first account is debited but the second account is not credited with the transferred amount. Also, the transaction processing system allows multiple transaction to run concurrently such that the database remains consistent before and after the transaction. The unit introduces basic properties of a transaction, the sequence in which operations of concurrent transactions are executed and the concept of recoverability that is, acceptable schedules from the point of transaction failure.

Space for learners:

2.2 LEARNING OBJECTIVES

After going through this unit, you will be able to:

- Explain basic concept of a transaction and its different states during execution.
- Tell the desirable properties of a transaction and why are these properties significant.
- Explain the concept of a schedule and concurrent execution of multiple transaction.
- Explain the how recoverability deals with issues that arise due to transaction failure during concurrent execution of transaction.

2.3 TRANSACTION CONCEPTS

A transaction is a logical unit of job which accesses and possibly changes the contents of a database. Transactions includes one or more database access operations such as insertion, deletion, modification and retrieval. A transaction can be embedded within an application program or can be specified using a high-level query language such as SQL. There can be several transactions in an application program separated by *begin* transaction and *end* transaction.

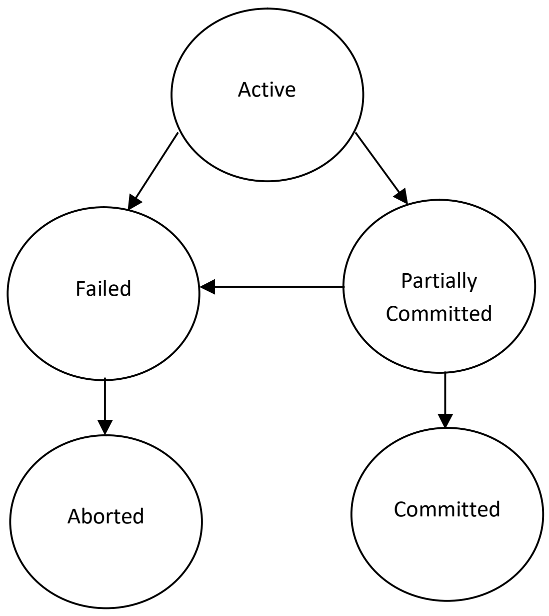


Figure 2.1: State Diagram of a transaction.

A transaction during the course of execution may go through several states as mentioned below. Figure 2.1 show the state diagram of a transaction during execution.

- Active – It is the initial state of any transaction during the period of execution.
- Partially Committed – It is the state when the transaction executes its last operation.
- Failed – It is a state when the transaction can no longer continue its normal execution.
- Aborted – It is the state when the transaction fails as a result of which it is rolled back that is the database is restored to the state earlier to the start of the transaction. At this point there are two possibilities either to restart the transaction or to kill the transaction. In case of software or hardware error, the transaction is restarted whereas in case of logical error in the code the transaction is killed.

Space for learners:

- Committed – It is the state when the transaction has executed all its operations successfully, that is all the changes are permanently stored on the database system.

Space for learners:

2.4 DESIRABLE PROPERTIES

A database management system ensures the integrity of data before and after the execution of the transaction by having the following four properties, which are also called ACID properties.

- Atomicity
- Consistency
- Isolation
- Durability

Atomicity: The term atomicity means that either all the operations in a transaction are executed completely or none of the operations are executed. There should not be any partial execution i.e. a situation where only few operations of the transaction are executed and the remaining operations are not executed. For example, consider X and Y are having INR 1200 and INR 1300 respectively in their bank account. X wants to transfer INR 200 from his account to the account of Y. So, we can consider it to be a transaction having six operations as shown below:

Transaction: *fund_transfer*

Operation 1	Read balance from the account of X
Operation 2	Subtract 200 from the account of X
Operation 3	Write updated balance in the account of X
Operation 4	Read balance from the account of Y
Operation 5	Add 200 from the account of Y
Operation 6	Write updated balance in the account of Y

Suppose a failure occurs during the execution of the transaction, *fund_transfer*. The failure can be due to several reasons such as hardware failure like hard disk crash or software error or power failure. If the failure occurs between operation 3 and operation 4 then in such a situation the amount 200 will be debited from the account of X whereas the amount 200 will not be credited to the account of Y. Thus the balance in account of X and Y will respectively INR 1000 and INR 1300. The amount INR 200 will be lost due to failure and to avoid such situation atomicity or execution of all operations must be ensured in the transaction.

Consistency: A transaction preserves the consistency of the database if, complete execution of the transaction changes the database from one consistent state to another consistent state. For example, consider the same transaction as discussed above in case of atomicity to transfer INR 200 from account of X to the account of Y.

It must be ensured that all operations in the transaction are executed completely to maintain the consistency before execution of the transaction and after the execution of the transaction. So, before the execution of the transaction sum of balance of account X and account Y is INR 2500 and after the completion of all the operations of the transaction the sum of balance of account X and account Y will still be INR 2500, which maintains the consistency of the database. However, it must be note that during the execution of the transaction, the database may be in inconsistent state. So, consistency of the database is always checked before the start of the transaction and after the completion of the transaction.

Isolation: The isolation property of a transaction is very important in context of concurrent execution of several transactions. If this property is not ensured it will result in inconsistency in database. Let us understand the property using an example.

Consider two transactions:
T1: Transfer of funds form account of X to account of Y
T2: Display total balance of account X and account Y

	T1	T2
Operation 1	Read balance of account of X	
Operation 2	Subtract 200 from the account of X	
Operation 3	Write updated balance in the account of X	
Operation 4		Read balance of account of X
Operation 5		Read balance of account of Y
Operation 6		Add both the Balance
Operation 7		Display total balance
Operation 8	Read balance of account of Y	
Operation 9	Add 200 from the account of Y	
Operation 10	Write updated balance in the account of Y	

Figure 2.2: Concurrent Transaction

Suppose transaction, T2 reads the balance in the account X and account Y sums it up and display's the total balance and all these operations are executed while transaction, T1 is in execution as shown in Figure 2.2. Even though both the transaction can complete their execution but the final result will be inconsistent in the case of transaction, T2. Considering X and Y are having INR 1200 and INR 1300 respectively in their bank account. Transaction, T2 will display the result as 1000 + 1300 = 2300 which is inconsistent and should have been 2500 instead. This inconsistency in result is due to the fact that T2 was executing before T1 could complete its execution.

One way of avoiding such inconsistency in case of concurrent execution of transaction is to execute them serially i.e. one after another. However, there are performance benefits of executing transaction concurrently. The isolation property of the transaction ensures that concurrent execution of transaction is equivalent to serial execution of the transactions in some order i.e. in concurrent execution of transaction one transaction is unaware that other

Space for learners:

transaction is executing. It looks as if both or multiple transaction are executed in isolation.

For a pair of transaction T1 & T2, it appears to T1 that T2 will start after T1 finishes its execution or T2 finished its execution before T1 started.

Durability: The durability property of a transaction ensures that after successful completion of a transaction, all the changes done to the database remains unchanged even if there is any hardware failure, software failure or power failure. Failure may result in data loss in main memory but data in disk are never lost. This is ensured by having the changes to data written to disk first before the transaction completes and also the information about the changes done to the database by the transaction that is the log file be written to disk, so that recovery from failure can be done when the system is restarts.

2.5 SCHEDULES

Transactions are set of operations performed on the database. So, in case of concurrent execution of multiple transaction, there is a requirement of a sequence in which the operations are executed because at a time only one operation can be performed on the database. This sequence of operations is known as Schedule. For example, Figure 2.3 shows a serial schedule where transaction P completes its execution before transaction Q can start.

To access and process data, a transaction needs to perform read and write operation. A read operation, *read (X)* reads the item X from database and stores it in local buffer. A write operation, *write (X)*, writes the item X from the local buffer to the database.

For example, consider the two transaction

P:
```
read(X)
read(Y)
X = X + 100
write(X)
```

Space for learners:

```
    Y = Y + X
    write(Y)

Q:  read(X)
    temp=X * 0.5
    X = X + temp
    write(X)
```

Let us assume that the initial value of X and Y are 100 and 500 respectively. In the schedule 1 shown in Figure 2.3 the two transaction P & Q are executed serially that is one after another. So, the final value of X and Y will be 300 and 700 respectively. If, however the order of execution of the transaction is reversed with Q executing before P as shown in schedule 2 of Figure 2.4 then the final value of X and Y will be 250 and 750 respectively.

Transaction: P	Transaction: Q
read(X)	
read(Y)	
X = X + 100	
write(X)	
Y = Y + X	
write(Y)	
	read(X)
	temp=X * 0.5
	X = X + temp
	write(X)

Figure 2.3: Schedule 1 - A serial schedule P followed by Q.

When several transactions are executed concurrently, then they need not be executed in serial order. The operating systems does a context switch between the transaction. That is the operating system will execute some operations of one transaction then switches to other transaction and executes some of its operations. The process of switching between the transactions cannot be decided on the number of operations that will be executed before switching. Figure 2.5 shows an example of concurrent schedule equivalent to schedule 1 that preserves the consistency of the database. It has to be noted that schedule 3 is one of the possible ways of executing transactions concurrently which preserves database consistency. The final value of X and Y will be 300 and 700 respectively for schedule 3. There are many ways of executing transaction concurrently, one such way is shown in Figure 2.6 of schedule 4. However, schedule 4 does not preserves database consistency as the final value of X and Y will be 150 and 700 respectively.

Transaction: P	Transaction: Q
	read(X)
	temp=X * 0.5
	X = X + temp
	write(X)
read(X)	
read(Y)	
X = X + 100	
write(X)	
Y = Y + X	
write(Y)	

Figure 2.4: Schedule 2 - A serial schedule Q followed by P.

Space for learners:

Transaction: P	Transaction: Q
read(X)	
read(Y)	
X = X + 100	
write(X)	
	read(X)
	temp=X * 0.5
	X = X + temp
	write(X)
Y = Y + X	
write(Y)	

Figure 2.5: Schedule 3 - A concurrent schedule equivalent to schedule 1.

Transaction: P	Transaction: Q
read(X)	
read(Y)	
	read(X)
	temp=X * 0.5
X = X + 100	
write(X)	
	X = X + temp
	write(X)
Y = Y + X	
write(Y)	

Figure 2.6: Schedule 4 - A concurrent schedule.

Space for learners:

2.6 RECOVERABILITY

The concept of recoverability deals with issues that arise due to transaction failure during concurrent execution of transaction. Suppose that two transactions T1 and T2 are executed concurrently. If there is some kind of dependency exists between the two transactions like the data produced by T1 using the write operation is consumed by T2 using read operation. In such case if the transaction T1 is aborted due to some failure, the dependency of T2 on T1 will result in aborting T2. In this context there are two types of schedules that are accepted from the point of recovery from transaction failure.

2.6.1 Recoverable Schedules

Recoverable schedule deals with the pair of transaction T1 and T2 such that if the data item produced by T1 is consumed by T2, then the commit operation of T1 must be executed before the commit operation of T2.

For example, in schedule 5 of Figure 2.7, the commit operation of transaction Q is executed before the transaction P. Now, if the transaction P fails before the commit operation then the data item X read by transaction Q is invalid and must be aborted. However, as the transaction Q has already committed, it cannot be aborted. Schedule 5 having commit operation by transaction Q before execution of transaction P makes it non recoverable schedule. Therefore, it must be ensured that all schedules are be recoverable.

Transaction: P	Transaction: Q
read(X)	
write(X)	
	read(X)
read(Y)	

Figure 2.7: Schedule 5.

Space for learners:

2.6.2 Cascadeless Schedule

Space for learners:

Consider the situation of schedule 6 in Figure 2.8. The data item X produced by transaction P is consumed by transaction Q. Similarly, the data item X produced by transaction Q is consumed by transaction R. Clearly, we could see that for data item X dependency of transaction R on transaction Q and the dependency of transaction Q on transaction P. Now, if there is a situation where transaction P fails then transaction Q and transaction R also required to be rolled back. This is called cascading rollback and is undesirable due to significant number of rollbacks required for recovery. So, it is preferred to avoid such schedules which may have cascading effect. Such schedules which avoids cascading rollback are called Cascadeless schedule.

Transaction: P	**Transaction: Q**	**Transaction: R**
read(X)		
write(X)		
	read(X)	
	write(X)	
		read(X)
		write(X)

Figure 2.8: Schedule 6.

CHECK YOUR PROGRESS

1. Which of the following represents the ACID properties of a transaction?
a) Atomicity, Consistency, Integrity, Durability
b) Atomicity, Concurrency, Isolation, Durability
c) Atomicity, Concurrency, Integrity, Durability
d) Atomicity, Consistency, Isolation, Durability

2. Which of the statement is true about transaction?

a) A program in execution
b) A logical unit of work
c) A set of operations
d) A set of operations to carry out a work.

3. Execution of a transaction in ________ preserves____________.
a) Atomicity, Consistency
b) Isolation, Atomicity
c) Isolation, Consistency
d) Consistency, Durability

4. Which of the following is not a transaction state?
a) Committed
b) Failed
c) Rollback
d) Aborted

5. What happens to a transaction in abort state when it is rolled back?
a) Kill or Restart
b) Kill
c) Restart
d) Allow new transaction

6. The ______ property of a transaction preserves the consistency of the database if complete execution of the transaction changes the database from one consistent state to another consistent state.
a) Atomicity
b) Isolation
c) Consistency
d) Durability

Space for learners:

7. The _______property of a transaction means either all the operations in a transaction are executed completely or none of the operations are executed.
a) Atomicity
b) Isolation
c) Consistency
d) Durability

8. _____________schedule deals with the pair of transaction such that if the data item produced by first transaction is consumed by second transaction then the commit operation of first transaction must be executed before the commit operation of second transaction.
a) Recoverable
b) Cascadeless
c) Cascading
d) Durable

9. To avoid cascading rollback, it is desirable to have ___________ schedule.
a) Recoverable
b) Cascadeless
c) Cascading
d) Durable

10. The ____________ property of a transaction ensures that after successful completion of a transaction, all the changes done to the database remains unchanged even if there is any failure.
a) Atomicity
b) Isolation
c) Consistency
d) Durability

Space for learners:

2.7 SUMMING UP

Space for learners:

- A transaction is a logical unit of job which accesses and possibly changes the contents of a database.
- A transaction during the course of execution may go through several states such as active, partially committed, Failed, Aborted and Committed.
- A transaction is said to be in Committed state when the transaction has executed all its operations successfully, and all the changes are permanently stored on the database system.
- A transaction is in Aborted state when the transaction fails as a result of which it is rolled back that is the database is restored to the state earlier to the start of the transaction.
- A database management system ensures the integrity of data before and after the execution of the transaction by having the following four properties, which are Atomicity, Consistency, Isolation and Durability.
- The term atomicity means that either all the operations in a transaction are executed completely or none of the operations are executed.
- A transaction is preserves the consistency of the database if complete execution of the transaction changes the database from one consistent state to another consistent state.
- The durability property of a transaction ensures that after successful completion of a transaction, all the changes done to the database remains unchanged even if there is any failure.
- The sequence of operations in case of concurrent execution of multiple transaction is known as Schedule.
- Recoverability deals with issues that arise due to transaction failure during concurrent execution of transaction.

2.8 ANSWERS TO CHECK YOUR PROGRESS

Space for learners:

1. d
2. d
3. c
4. c
5. a
6. c
7. a
8. a
9. b
10. d

2.9 POSSIBLE QUESTIONS

1. List the ACID properties of a transaction. What is the significance of these properties?
2. A transaction goes through several states before it commits or aborts. Explain all these states and its importance.
3. What is a transaction? Does serial execution of two transaction equivalent to concurrent execution of the same two transactions.
4. What are the benefits of concurrent execution of transactions?
5. Define a schedule with a proper example.
6. Explain the isolation property of a transaction.
7. Write a schedule that suffers from cascading rollback and a schedule that do not suffer from cascading rollback.
8. When a transaction enters abort state, what are the actions taken by the database system.
9. What is a recoverable schedule?
10. What is a Cascadeless schedule? Explain using an example.
11. Consider two transaction T1 and T2. Does the order of execution affect the final result of the common data items involved in the execution?

12. Consider the transaction given below with initial values of X and Y as 100 and 200. What will be the final value of X and Y if the transactions are executed in the order:

    i. T1 followed by T2

    ii. T2 followed by T1

```
T1:   read(X)                T2:   read(X)
      X = X * 10                   X = X + 20
      write(X)                     write(X)
      read(Y)
      temp=X+X*0.2
      Y = Y + temp
      write(Y)
```

Space for learners:

2.10 REFERENCES AND SUGGESTED READINGS

- Database System Concepts 6th Edition by Abraham Silberschatz, Henry F. Korth, S. Sudarshan, McGraw – Hill International Edition.
- Fundamentals of Database System Seventh Edition, by Elmasri Ramez and Navathe Shamkant, Pearson.
- Database Management Systems by Raghu Ramakrishnan, Johannes Gehrke, Irwin Computer Science.
- Database Systems: The Complete Book by Hector Garcia-Molina, Jeffrey Ullman, Jennifer Widom.

UNIT 3: CONCURRENCY CONTROLAND RECOVERY TECHNIQUES

Space for learners:

Unit Structure:

3.1 Introduction

3.2 Objective

3.3 Concurrency

3.3.1 Requirement of Concurrency

3.4 Transaction and Transactional Properties

3.4.1 States of Transaction

3.4.2 Transactional Properties

3.5 Schedule of Transaction

3.6 Serial, Non-serial, Conflict-Serializable Schedule

3.7 Concurrency Controls

3.7.1 Lock and Modes of Locking

3.7.2 Lock Compatibility

3.7.3 Two-Phase Locking Techniques for Concurrency Controls

3.7.4 Deadlock

3.8 Recovery of DBMS

3.9 Transaction Failure

3.10 Recovery System in DBMS for Transaction Failure

3.10.1 Log-based Recovery

3.10.2 Shadow Paging

3.11 Summing Up

3.12 Answers to Check your Progress

3.13 Possible Questions

3.14References and Suggested Readings

3.1 INTRODUCTION

Space for learners:

Banking industry, online shopping, stock markets, airline reservation, IRCTC reservation required transaction processing system which allows large databases to access and thousands of concurrent users perform operations on database transactions concurrently. The system that is employed for such tasks should be of high availability and faster response time to cope with hundreds and thousands of concurrent users. A transaction is a logical unit of database processing in which it includes commands such as retrievals, insertion, update and deletion.

3.2 UNIT OBJECTIVES

The unit is describing the concurrency control and recovery techniques in database transactions. After completing the unit students′ will able to:

- Understand the requirement of Concurrency Controls,
- Describe the States of Transaction,
- Explanation of Transactional Properties,
- Detailed discussion on Schedules of Transaction,
- Concurrency Controls,
- Learn about Locks and Locking and lastly why Recovery is necessary.

3.3 CONCURRENCY

A database which is used by many users, access the data concurrently. However, a single user database management system is limited to personal computers only; most database management systems are multi-user. The concept of multiprogramming allows the operating system (OS) to execute multiple processes at the same time such that multiple users can access the database simultaneously. A single CPU executes only a single process at a particular time. While in case a multiprogramming OS, it executes a process then halt the process and execute the next process, so on and so forth. A process which is halted earlier is resumed at an instance where it was suspended whenever CPU processing time is given to it. This concurrent execution of processes is interleaved which means when a process is in the CPU and waiting for

Input or Output (I/O) operation, the CPU time is shifted to another process that way the CPU is always kept busy. Suppose, there are two processes P_1 and P_2executing concurrently in an interleaved manner. Process P_1 is waiting for I/O operation, process P_2 will be executed which was waiting for CPU time such that interleaved method doesn't allow the CPU to be idle while P_1 is waiting for I/O time. Benefit of interleave is that it doesn't allow a long process to delay other processes. In DBMS, the primary resources are the stored data in the database that can be accessed concurrently by many users which allows the users to retrieve and modify the database concurrently.

Space for learners:

STOP TO CONSIDER

A multiprogramming operating system executes multiple processes concurrently similarly using a multiprogramming operating system multiple transactions of DBMS are achieved concurrently.

3.3.1 Requirement of Concurrency Control

Concurrency control and recovery mechanisms are deployed on database as operations related to transactions. Various users may submit transactions which are executed concurrently to access and update the database. If uncontrolled concurrent transactions are executed it may lead to many issues such as an inconsistent database. Let us discuss some of the problems by taking a Railway Ticket Reservation System.

READ (A): Reads a database named A into a variable in a program.

WRITE (A): Writes the value of variable A into the database item named A.

T_1

READ (A);
A = A - X;
WRITE (A);
READ (B);
B = B + N;

Figure 3.0 a) Transaction T_1

Space for learners:

T_2

READ (A);
A = A + R;
WRITE (A);

Figure 3.0 b) Transaction T_2

Figure 3.0 (a) Transfer of X reserved seats from one train which is stored in database item named A to another train whose reserved seats are stored in database item named B in transaction T_1. Figure 3.0 (b) T_2 transaction implies to reserved R seats which refers to transaction in T_1 that is A.
The same program can be used for multiple transactions as each of the transactions will have a different date, number of reserved seats and train number. In figure 3.0 a) and b) there are T_1 and T_2 transaction have specific date, seat and train number which are stored in A and B database that is the purpose of concurrency control. If these two transactions are executed concurrently, we may encounter different types of problems which are discussed below

- **Temporary Update or Dirty Read Problem.** This issue occurs when a transaction makes changes in the database item, then the transaction fails due to some error. At the same time, the updated item from the database is read by another transaction before it could be changed back to its original value in the database.

T_1	T_2

READ (A);
A = A - X;
WRITE (A);	

	READ (A);
	A = A + R;
	WRITE (A);

READ (Y);	

Figure 3.1 Temporary Update

For example, in Figure 3.1 shows an item A is updated by T_1 transaction and suppose T_1 fails before the transaction completion. As the transaction fails to complete the system should change A to the original value in the database. Transaction T_2 access the value of A which becomes a temporary value before the system could update it in the database and the value of A is not recorded permanently in the database. As this temporary value of A is accessed by T_2 transaction that is why this problem is called dirty read.

- **Incorrect Summary Problem.** When a transaction is processing a number of database items by calculating an aggregate summary function and another transaction is updating these items. While the aggregate function may calculate few values before and after updation of the database items.

T_1	T_2
SUM = 0;	
READ (X);	
SUM = SUM +X;	
●	READ (A);
●	A = A - R;
●	WRITE (A);
READ (A);	
SUM = SUM + A;	
READ (B);	
SUM = SUM + B;	
	READ (B);
	B = B + R;
	WRITE (B);

Figure 3.2 Incorrect Summary Problem

For example, T_1 transaction is executing and calculating number of reserved seats in a train as shown in Figure 3.2. The transactions are running in an interleaved manner. The resultant value of T_1 will be incorrect by an amount X as T_1 access the

Space for learners:

value of A once X seats are subtracted from A and the value of B is accessed before the reserved seats X are added to it.

- **Loss Update Problem.** This kind of issue occurs when two transactions access the same item of the database while their operations are interleaved which makes mistakes in values of some items and hence makes the database inconsistent.

T_1	T_2
READ (A);	
A = A - X;	
	READ (A);
	A = A + R;
WRITE (A);	
WRITE (B);	
	WRITE (A);
B = B + X;	
WRITE (B);	

Figure 3.3 Loss Update Problem

We presume that the transactions T_1 and T_2 are executed at the same time and they are processed in a interleaved manner. In Figure 3.3, the final result of A is incorrect as T_2 reads the value of X, update it before T_1 changes it in the database. Value of A is overwritten in T_2 and hence the updated value of T_1 is lost.

- **Unrepeatable Read Problem.** This problem occurs when two or more reading variables of the same transaction tries to access different values of the same variable. As transaction T_1 read some value twice during a single transaction and the value of the item is changed by T_2 transaction in between the read commands. That is why it is called unrepeatable read as it gets different values of the same item.

Space for learners:

CHECK YOUR PROGRESS-I

1. What is concurrency in DBMS?
2. What is a multiprogramming operating system?
3. How are processes executed in an interleaved manner?
4. Why are concurrency controls required?
5. What kind of problem occurs when two transactions access the same item of the database while their operations are interleaved which makes mistakes in values of some items and hence makes the database inconsistent?

Space for learners:

3.4TRANSACTION AND TRANSACTIONAL PROPERTIES

The operations on a database in a form of transaction can be done by a user interface program or through a query language such as SQL. To form a transaction, it is to specify transaction boundaries which are BEGIN TRANSACTION and END TRANSACTION statements in an application program. The access operation of all databases between these two boundaries are considered as one transaction. To contain more than one transaction in a single program, it has to contain several transaction boundaries.

Read-Only Transaction. The program to retrieve only data and not to update the database in a transaction.

Read-Write Transaction. The program to retrieve and update the database in a transaction.

3.4.1 States of Transaction

A transaction is an atomic unit which means if a transaction is executed it should be completed entirely or not at all. For the purpose of recovery, the system needs to monitor each transaction starts, terminate and abort. The recovery system should monitor the following

Begin_Transaction. Keeps the track of execution of the beginning of a transaction.

READ or WRITE. Read and write operation which is executed on a database as a part transaction.

End_Transaction. This monitors the operation of READ and WRITE have ended which is the end of transaction.

Commit_Transaction. It means a successful execution of a transaction, any changes executed by the transaction can be committed to the database and will not be changed.

RollBack or Abort. It means the transaction is unsuccessfully ended and the changes made by the transaction to the database must be ROLLBACK or undone.
The execution states of a transaction as follows:

A transaction starts, it moves from inactive to active state and it executes READ and WRITE operation. It enters a partially committed state once the transaction is completed. This activates some recovery protocols at this time, if the system fails it will not result in loss of data permanently. After the recovery protocol is executed successfully the transaction enters the committed state. In the committed state it is concluded that the execution is completed and the data are permanently recorded in the database, even if the system encounters a failure.

While a transaction can enter a fail state, if any failure of the checks is noticed or during the active state the transaction is aborted. Then the transaction may have to roll back the WRITE operation which is executed on the database. The transaction is halted due to the error corresponding to the terminated state**.**

3.4.2 Transactional Properties

Transactions should have four properties which are popularly known as ACID properties; these properties should be employed by the concurrency control and recovery mechanism of the database. The ACID is an abbreviation of Atomicity, Consistency, Isolation and Durability. These are as follows:

Atomicity: A transaction should be atomic in nature; it should either be completely executed or not at all.

Space for learners:

Consistency: A transaction is completely executed from beginning to end without any interference of other transactions; it means a transaction is consistency preserving. A consistent transaction will take the database from one state of consistency to another.

Isolation: A transaction should look like it is being run in isolation without interference from other transactions, even if many transactions are running concurrently.

Durability: The updates or changes executed on the database items by a committed transaction must be persistent. The changes (update) should not be lost due to any failure in the system.

Space for learners:

3.5 SCHEDULES OF TRANSACTIONS

A schedule of transactions is an order list of transaction such as *S* schedule of *m* transactions will be T_1, T_2, T_3, ……, T_m. In a schedule, *S* operations of different transactions can be interleaved. The transactions in the schedule *S* should be in the same order as they are in T_i. Schedule *S* operations are considered to be in total order which means one operation may execute before another from any two operations in the schedule.
In this topic we are interested in recovery and concurrency controls so our objectives are READ, WRITE, commit and abort operation. Here we used a few shorthand notations for begin_transaction is b, READ is r, WRITE is w, end_transaction is e, commit is c and abort is a such that

S_1: $r_1(A)$; $r_2(A)$; $w_1(A)$; $r_1(B)$; $w_2(A)$; $w_1(B)$;

S_2: $r_1(A)$; $w_1(A)$; $r_2(A)$; $w_2(A)$; $r_1(B)$; a_1;

To be in conflict the two operations in a schedule have to satisfy the three following conditions

1. Operations belong to different transactions.
2. Operations access the same item A.
3. Minimum one of the operations is WRITE.

In schedule S_1, operations $r_1(A)$ and $w_1(A)$ conflict. Similarly operations $r_2(A)$ and $w_1(A)$ and operation $r_1(A)$ and $r_2(A)$ do not conflict as they are read operations. Operations $w_2(A)$ and $w_1(B)$ also do not conflict as w_2 is on A and w_1 operation is on B. Also, the operations $r_1(A)$ and

$w_1(A)$ do not conflict as both the operations belongs to the same transaction.

READ-WRITE Conflict. The order of two operation are changed such as from $r_1(A)$; $w_2(A)$ to $w_2(A)$; $r_1(A)$, this changes in the transaction T_1 as it read the value of A as in the second order, the value of A is updated by $w_2(A)$ before it can be read by $r_1(A)$.

WRITE-WRITE Conflict. If the order of two operation is changed from $w_1(A)$; $w_2(A)$ to $w_2(A)$; $w_1(A)$. In write-write conflict the final value of A will differ as in first case it is changed by T_2 and later by T_1. It is to be noticed that two read operations do not conflict by changing their order which makes no difference in the final result.

Schedules depend on serializability. The schedules which are always observed to be correct when executed concurrently are known as serializable schedules. Suppose a DBMS transaction is submitted with T_1 and T_2 simultaneously. If no interleaved operation is allowed; it will lead to two possible results.

A. In transaction T_1 all the operation will be executed in sequence followed by transaction T_2 in sequence.
B. All operations of transaction T_2 will be executed followed transaction T_1 in sequence such transactions are called serial schedule.

3.6SERIAl, NON-SERIAL and CONFLICT-SERIALIZABLE SCHEDULE

T_1	T_2

READ (A);
A = A - R;
WRITE (A);
READ (B);
B = B + R;
WRITE (B);	

	READ (A);
	A = A + O;

Space for learners:

	WRITE (A);

Figure 3.5 (a) Schedule 1

T$_1$	T$_2$

	READ (A);
	A = A + O;
	WRITE (A);

READ (A);
A = A - R;
WRITE (A);
READ (B);
B = B + R;
WRITE (B);	

Figure 3.5 (b) Schedule 2

T$_1$	T$_2$

READ (A);
A = A - R;	

	READ (A);
	A = A + O;

WRITE (A);
READ (B);	

	WRITE (A);

B = B + R;
WRITE (B);	

Figure 3.5 (c) Schedule 3

T$_1$	T$_2$

Space for learners:

3.5

READ (A);
A = A - R;
WRITE (A);	

	READ (A);
	A = A + O;
	WRITE (A);

READ (B);
B = B + R;
WRITE (B);	

Figure (d)

Schedule 4

Schedule 1 in Figure 3.5 (a) and Schedule 2 in Figure 3.5 (b) are serial as the operations of each transaction are executed without interleaved operation from other transactions. In a serial schedule, transaction are executed serially such that in the order of T_1 and then T_2 and T_2 and then T_1.

Schedule 3 in Figure 3.5 (c) and Schedule 4 in Figure 3.5 (d) each from the two transactions each sequence interleaves operation so they are called non-serial schedules. Initializing database items with the value A = 80, B = 80 and R = 3 and O = 2. After transactions, T_1 and T_2 are executed. We expected the database values to be A = 79 and B = 83 according to the layout of the transaction. Schedule 1 or Schedule 2 gives a correct outcome as they are executed in a serial schedule. Considering non-serial Schedule 3 and Schedule 4 for execution. Schedule 3 gives outcome A = 82 and B = 83, where the value of A is wrong and Schedule 4 gives a correct outcome.

The outcome of Schedule 3 is wrong due to the lost update problem. While some non-serial schedules are calculated correctly such as Schedule 4. We would like to know which gives the wrong result and which gives the correct result. This is used to characterize schedules in the fashion of serializability of a schedule.

By the definition of serializable schedule: A schedule S of m transactions is serializability when it is equivalent to a few serial schedules of the same m transaction. By implying a non-serial schedule is serializable it means the result is correct that is the same as in a serial schedule which is regarded as correct. Two schedules that produce the same final outcome of the database are called result equivalent. The schedule to be equivalent to the execution done on each item in one

Space for learners:

schedule should also be applied to database items in both the schedules in similar order.

If the order of any two conflicting operations is the same it is said to be the conflicting equivalent, if conflict equivalent some schedule S` and it can be reordered the non-conflicting operation in S till it forms the equivalent series of schedule S` is said to be conflict serializable.

STOP TO CONSIDER

- Transactions should have four properties which are popularly known as ACID properties; these properties should be employed by the concurrency control and recovery mechanism of the database. The ACID is an abbreviation of Actomicity, Consistency, Isolation and Durability.
- The schedules which are always observed to be correct when executed concurrently are known as serializable schedules

CHECK YOUR PROGRESS-II

6. What do you mean by transaction in DBMS?
7. Define Read- only transaction?
8. Define Write-only transaction?
9. Why is a transaction required to be atomic?
10. What are the states of a transaction?

3.7 CONCURRENCY CONTROLS

3.7.1 Lock and Modes of Locking

A variable associated with each data that implies whether a read operation or a write operation is to be implemented to the data items is called lock. The specialty of lock is that it gives synchronized access to the data items by concurrent transactions.

Space for learners:

The concurrency control technique which allows the locked data items to be accessed and manipulated is called locking, to maintain the consistency and integrity of the database. DBMS implements two modes of locking namely Exclusive and Shared.

Exclusive Lock. It provides exclusive controls on the data item to a transaction. The transaction must acquire the exclusive lock to read and to write a data item. Hence, an exclusive lock is also known as update lock or write lock. For example, a transaction T_1 acquires exclusive lock on data item P. No other transaction is allowed to access P until transaction T_1 releases its lock on P.

Shared Lock. When a transaction wants to read a data item only, not to modify it in such instances, shared lock can be implemented on that data item. It is also known as read lock. For example, a transaction T_1 has acquired a shared lock on data item P, T_1 can read P but cannot write on P. Furthermore, multiple transactions can acquire shared locks on P at the same time. However, any transaction can acquire an exclusive lock on P.

3.7.2 Lock Compatibility

When a transaction requires to perform some operation on a data item, it requests for an appropriate lock mode on the data item. The lock manager grants lock immediately if the requested data item is not locked by any other transaction. Otherwise, the lock request may or may not grant depending on the compatibility of locks. Lock compatibility regulates whether locks can be acquired on a data item by any number of transactions simultaneously. For example, a transaction T_1 request a lock of mode m_1 on a data item P on which another T_2 transaction currently holding a lock of mode m_2. If mode m_1 and m_2 are compatible the request will be grant immediately; otherwise rejected. The lock compatibility matrix:

Requested Mode	**Shared**	**Exclusive**
Shared	YES	NO

Space for learners:

Exclusive	NO	NO

Figure 3.6 Compatibility matrix

Space for learners:

In figure 3.6 the term 'YES' indicates the request can be granted and the term 'NO' indicates request cannot be granted.
The lock request of transaction T_1 is granted immediately if m_1 is shared and, if and only if m_2 is also shared. Otherwise the lock request is not granted and transaction T_1 has to wait. Furthermore, if mode $m_1$is exclusive, then the lock request by transaction T_1 will not be granted and T_1 has to wait.

3.7.3 Two- Phase Locking Techniques for Concurrency Control

Two phase locking 2PL is a lock-based concurrency control technique that divides each transaction into two phases. At the first phase, the transaction acquires all the locks and during the second phase it releases all the locks.

- **Grow Phase.** In this phase the number of locks held by a transaction increases from zero to maximum.
- **Shrinking Phase.** During this phase the locks are released, due to which it is called the shrinking phase. The number of locks held by the transaction decreases from maximum to zero.

Whenever a transaction releases a lock data item it enters the shrinking phase. Once a data item in the shrinking phase, it is not allowed to acquire any locks further. Therefore, the release of locks must be delayed until all the required locks on the data items are acquired. Considering two transactions T_1 and T_2 along with their lock request.

T_1

LOCK-X (A);
READ (A);
A = A - 200;
WRITE (A);
UNLOCK (A);
LOCK-X (P);

READ (P);
P = P + 200;
WRITE (P);
UNLOCK (A);

T_2

LOCK-X (SUM);
SUM = 0;
LOCK = 0;
LOCK -S (P);
READ (P);
SUM = SUM + P;
UNLOCK (P);
LOCK-S (A);
READ (A);
SUM = SUM + A;
WRITE (SUM);
UNLOCK (A);
UNLOCK (SUM);

Figure 3.7 Transaction T_1 and T_2 with their lock request

T_1	T_2

LOCK-X (A);
READ (A);
A = A - 200;
WRITE (A);
UNLOCK (A);
LOCK-X (P);
READ (P);
P = P + 200;
WRITE (P);
UNLOCK (A);
	LOCK-X (SUM);
	SUM = 0;
	LOCK = 0;
	LOCK -S (P);
	READ (P);
	SUM = SUM + P;
	UNLOCK (P);
	LOCK-S (A);
	READ (A);
	SUM = SUM + A;
	WRITE (SUM);

Figure 3.8 Transaction T_1 and T_2 in two-phase locking

In figure 3.8 the statement for releasing the lock is written at the end of the transaction. But, such statement do not needs to appear at the end of

Space for learners:

the transaction to retain two-phase locking properties such as the UNLOCK (A) statement of T_1 may appear just after the LOCK- X(P) statement and still maintain the two phase locking property.

3.7.4 Deadlock

Deadlock occurs when all transactions in a set of two or more transactions are waiting for some data item that is locked by some other transaction. Hence, each transaction in the set is waiting for the other waiting transaction in the set. None of the transactions can proceed until and unless one of the waiting transactions releases lock on the data item. Deadlock can be prevented by applying deadlock prevention protocols.

3.8 RECOVERY OF DBMS

When a transaction of DBMS is executed, it is the responsibility of the system to make sure all the operations in a transaction are completed successfully and their outcomes are recorded in the database permanently or the transaction has no effect on the database. If the transaction fails while executing it shouldn't have any effect on the database.
Types of failure may encounter which executing DBMS in the middle of a transaction are:

1. **A System Crash.** During the execution of the transaction there could be a hardware, software or network error. Hardware crash mainly consists of main memory failure.
2. **System error.** The transaction may cause failure due to the programming error which may be caused by various reasons such as division by zero or integer overflow. Furthermore, the user may also interrupt the execution of the transaction in between.
3. **Concurrency Control Enforcement.** A transaction can be aborted by concurrency control method due to the violation of serializability or it may also abort transaction or more transactions to avoid deadlock among several transactions. Transactions aborted due to these kinds of reasons are started automatically later.

Space for learners:

4. **Exception Conditions or local detected by the transaction.** During the execution of the transaction, many necessary conditions need to be fulfilled such as data for a transaction may not be found.
5. **Disk Failure.** There can be a disk head crash during read or write operation or some disk block may lose their data due to a read or write malfunction. Such events can happen during a read or write operation of a transaction.
6. **Physical Damage and Catastrophes.** This is a never ending list of problems which includes power failure, fire, theft, over writing disk by mistake, damaging the hardware by physical means etc.

3.9 TRANSACTION FAILURE

A transaction aborts when it is failed to execute or the transaction reaches an instance where it can't go any further. The atomicity property of transaction, which suggested that all operations in a transaction have to execute completely or not at all. There cannot be a case where only half of the operation will be executed or else this will lead to a transaction failure. Reason for transactions failure:

Logical error: This kind of error occurs when a transaction cannot complete due to some code errors or any internal condition error.

System error: In this kind of error, when the database system terminates an active execution of a transaction as the system is not able to execute it. This kind of error occurs due to deadlock or unavailability of resources, the system aborts an active transaction.

3.10RECOVERYSYSTEMIN DBMS FORTRANSACTION FAILURE

There are two techniques to recover if the system encounters a transaction failure. They are

1. Log-based recovery
2. Shadow paging

3.10.1 Log-based recovery

Space for learners:

A log is a sequence of records that maintains the history of all the updates which are implemented on the database. It used to hold the records of modification done on the database and it is also known as system log. In an ongoing transaction, if the system crashes, by using log files it can be returned back to its previous state as if nothing has happened to the database.

Suppose,

< T_1, X_1, D_1, D_2 > : Updates log records where, T_1 is the transaction, X_1 is the data, D_1 is the previous data and D_2 is the new data.

< T_1, starts > : Transaction T_1 start executing

< T_1, Commit > : Transaction T_1 to commit

< T_1, Abort > : Transaction T_1 is aborted

Deferred Database Modification

In deferred database modification the updation is delayed or deferred in the database until the last operation of a transaction is executed and reaches to completion.

Recovery system:

Redo (T_1) : All data items updated by the transaction T_1 are set to a new value.

Immediate Database Modification

In this type of modification, the database is modified after a WRITE operation, it immediately modifies the database whenever a transaction performs a WRITE or update operation. Update log records contain both previous and new values of data items.

Recovery system:

Undo (T_1) : All the data items changed by T_1 transaction are set to the previous values.

Redo (T_1) : All the data items changed by T_1 transaction are set to new values.

3.10.2 Shadow Paging

Shadow paging is an alternative to log-based recovery. In shadow paging it maintains two tables during the execution period of a transaction; a current table and a shadow table. The shadow page table is stored in non-volatile memory, such that the state of the database prior to transaction execution can be removed and the shadow page table is never modified during execution.

Both the page tables are identical where the current page table is accessing used data items during the execution of the transactions. Wherever any page is written for the first time, a copy of this page is made on an inside page. The current page table is then made to point to the copy and the updation is performed on the copy

Space for learners:

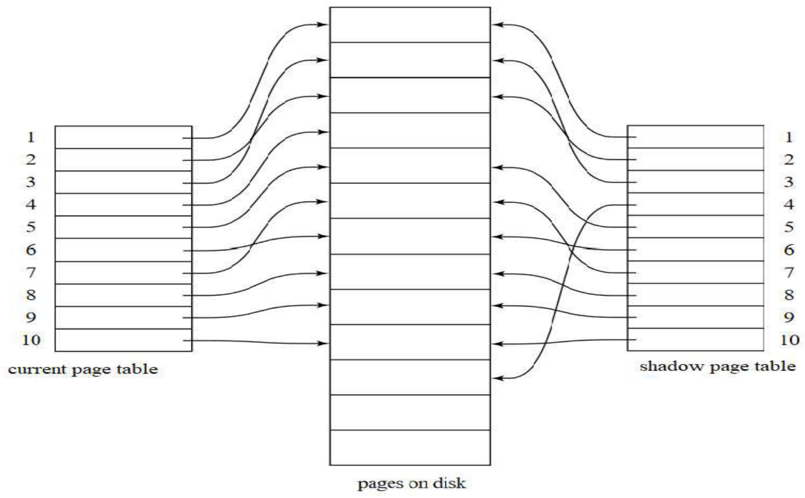


Figure 3.9 :Shadow and current page table

To commit a transaction

1. Put all modified pages in main memory to disk
2. Put all output current page table to disk.
3. Make the current page, the new shadow page. Keeps a pointer to the shadow page table at a fixed known location on disk.

Once the pointer to the shadow page table has been written, a transaction is committed. If there is a crash encountered no recovery is required. After a crash a new transaction can start immediately using the shadow page table.

Space for learners:

CHECK YOUR PROGRESS-III

Multiple Choice Questions

11. When multiple transactions are processed in a controlled manner using some protocols is known as
    (a) Data Control
    (b) Entity Control
    (c) Concurrency Control
    (d) DBMS Control

12. The concurrent execution of processing is interleaved transaction will be executed in
    (a) T_1 then T_2 and T_2 then T_1
    (b) T_1 then T_4 and T_2 then T_1
    (c) T_9 then T_5 and T_2 then T_1
    (d) None of these

13. When two transactions access the same item of the database while their operations are interleaved which makes mistakes in values of some items and hence makes the database inconsistent this kind of problem is known as
    (a) Loss Update Problem
    (b) Dirty Read Problem
    (c) Unrepeatable Read Problem
    (d) Incorrect Summary

14. During the process of transaction, the recovery system should monitor
    (a) Begin Transaction
    (b) Read Write
    (c) Commit Transaction
    (d) All of these above

15. When a transaction is unsuccessfully ended the changes made by the transaction to the database must be
    (a) Read
    (b) Write
    (c) Rollback or undone
    (d) All of the above

16. As transaction T_1 read some value twice during a single transaction and the value of the item is changed by T_2 transaction in between the read commands. That is called
    (a) Temporary Update Problem
    (b) Unrepeatable Read Problem
    (c) Loss Update Problem
    (d) Incorrect Summary Problem
17. Transactional properties should have four properties popularly known as
    (a) Read - write property
    (b) Write - read property
    (c) ACID property
    (d) BASE property

18. The concurrency control technique allows to lock data items to be
    (a) Accessed and manipulate
    (b) Extract information
    (c) Enter and Access
    (d) Process Information
19. How many locking modes are there
    (a) One
    (b) Two
    (c) Three
    (d) Four
20. Which incident may cause physical damage to the system
    (a) Earthquake
    (b) Hacking
    (c) Virus
    (d) None of these

Space for learners:

3.11 SUMMING UP

- In DBMS, the primary resources are the stored data in the database that can be accessed concurrently by multiuser which allows the user to retrieve and modify the database concurrently.

- Concurrency control and recovery mechanisms are deployed on database operations in a transaction.
- The operations on a database in a form of transaction can be done by a user interface program or through a query language such as SQL.
- A transaction is an atomic unit, i.e., if a transaction is executed it should be completed entirely or not at all.
- For recovery, the system needs to monitor each transaction starts, terminate and abort.
- The ACID (properties of a Transaction) is an abbreviation of Atomicity, Consistency, Isolation and Durability.
- A schedule of transactions is an order list of transaction such as *S* schedule of *m* transactions will be T_1, T_2, T_3, ……, T_m.
- To be in conflict the two operations in a schedule have to satisfy the three following conditions
  - Operations belong to different transactions.
  - Operations access the same item A.
  - Minimum one of the operations is WRITE.
- Lock is a variable associated with each data that implies whether a read operation or a write operation is to be implemented to the data items.
- DBMS implements two modes of locking namely Exclusive and Shared.
- Deadlock occurs when all transactions in a set of two or more transactions are waiting for some data item that is locked by some other transaction.
- There are two techniques to recover if the system encounters a transaction failure. They are
  - Log-based recovery
  - Shadow paging

3.12 ANSWERS TO CHECK YOUR PROGRESS

1. Concurrency in DBMS is the process of storing data in the database that can be accessed concurrently by multiuser which allows the user to retrieve and modify the database concurrently.
2. OS executes a process then halt the process and execute the next process, so on and so forth. A process which is halted earlier is

resumed at an instance where it was suspended whenever CPU processing time is given to it; this is called Multiprogramming OS.

3. The concurrent execution of processes is interleaved which means when a process is in the CPU and waiting for Input or Output (I/O) operation, the CPU time is shifted to another process that way the CPU is always kept busy.
4. Concurrency control and recovery mechanisms are deployed on database operations in a transaction. Various users may submit transactions which are executed concurrently to access and update the database items. If uncontrolled concurrent transactions are executed it may lead to many issues such as an inconsistent database.
5. Loss Update is the kind of problem that occurs when two transactions access the same item of the database while their operations are interleaved which makes mistakes in values of some items and hence makes the database inconsistent.
6. A transaction is a logical unit of database processing in which it includes commands such as retrievals, insertion, update and deletion.
7. Read-Only Transaction is a program to retrieve only data and not to update the database in a transaction.
8. Read-Write Transaction is a program to retrieve and update the database in a transaction.
9. A transaction needs to be an atomic unit which means if a transaction is executed it should be completed entirely or not at all to make the database consistent
10. The states of transactions are Begin Transaction, READ or WRITE. End Transaction, Commit Transaction and Roll Back or Abort.
11. (c)
12. (a)
13. (a)
14. (d)
15. (a)
16. (b)
17. (c)
18. (a)

Space for learners:

19.(b)
20.(a)

3.13 POSSIBLE QUESTIONS

Short answers type questions

1. Why are concurrency controls required?
2. Explains the phases of transactional states and its execution process.
3. Discuss the desirable properties of transactions.
4. What are the necessary conditions to be fulfilled by a transaction to be in conflict?
5. What will be the possible outcome for two transactions submitted simultaneously for execution and if no interleaved operation is allowed?
6. What is serial schedule?
7. What is non-serial schedule?
8. What is conflict schedule?
9. What is locking?
10. Short note on Deadlock.

Long answers type questions

1. List a few database applications where transaction processing is used. List a few different transaction types for each application.
2. How concurrency is achieved using multiprocessing OS? What is the advantage of interleaved processing?
3. What are the problems occur when two transactions are executed in an uncontrolled manner? Give examples and explain.
4. Discuss the actions taken by READ and WRITE operations on a database.
5. Discuss the typical state that a transaction goes through during execution.
6. Discuss the atomicity, durability, isolation and consistency prevention.
7. What is serializable schedule? Why is a serial schedule considered correct? Why is a serializable schedule considered correct?
8. What is lock? Explained the modes of locking.

Space for learners:

9. What do you understand by lock compatibility? Discuss the details with examples.
10. Discuss the different types of failures. What is meant by catastrophic failure?

3.14 REFERECES AND SUGGESTED READINGS

1. Elmasri, Ramez, Navathe, Shamkant B., *Fundamentals of Database Systems,* Pearson Education.
2. Gehrke, Johannes, Ramakrishnan, Raghu, *Database Management Systems,* McGraw-Hill Education.

Space for learners:

UNIT 4: DATABASE SECURITY

Space for learners:

Unit Structure:

4.1. Introduction

4.2. Unit Objectives

4.3. Security issues

4.4. Principles of database security:

4.5. Security models

4.6. Some common threats in database security

4.7. Challenges of database security in DBMS

4.8. Control methods of database security

4.9. Multilevel security

4.10. Types of access control

4.10.1. Mandatory access control (MAC)

4.10.2. Discretionary access control (DAC)

4.10.3. Role-Based access control (RBAC)

4.11. Authentication and Authorization

4.12. Encryption

4.12.1. Data Encryption

4.12.2. Algorithms for Encryption Process

4.12.3. Disadvantages of encryption

4.13. Digital signature

4.14. Summing up

4.15. Answers to Check Your Progress

4.16. Possible Questions

4.17. References and Suggested Readings

4.1 INTRODUCTION

Space for learners:

The use of various tools to protect huge virtual data storage units is known as database security. The field consists of several components, but it is primarily concerned with how to protect user databases from external attacks. Protecting the data itself (data level security), the applications used to process and store data, the physical servers, and even the network connections that allow users to access databases are all areas of database security (system level security). Database security procedures protect the data within the database and the database management system, and all applications that access it from intrusion, data misuse, and damage. It's a large term that refers to various processes, tools, and methodologies used to ensure database security.

Here in this unit, we will discuss the various aspect of database security. You'll be able to understand the fundamental problems that compromise database security. As you'll see, database-driven systems can have problems at any stage, including during database creation, deployment, and even later. This unit aims to provide a brief overview of database security threats and various challenges. When developing a security system, security models are the most fundamental theoretical tool to use. Security policies, which are governing regulations adopted by any organization, are enforced by these models. Access control models are security models that restrict the activities of authorized users. Discretionary, mandatory, and role-based access controls are the three main types of access control. Every technique has its own set of advantages and disadvantages. The choice of an appropriate access control model is based on the requirements as well as the types of attacks that the system is vulnerable. A strong authentication and authorization strategy helps protect the users and their data from attackers. You will get a brief overview of the data encryption process. Different types of encryption algorithms will discuss at the last of this unit.

4.2 UNIT OBJECTIVES

After going through this unit, you will be able to:

- understand the needs of Implicit parallelism techniques.
- understand the database security issues
- know the principle of database security

- describe the security model
- describe the various control method of database security
- know different threats and challenges in database security
- know the idea of Multilevel security
- describe different types of access control mechanisms
- distinguish between authentication and authorization
- understand the encryption process and its various types of algorithm
- know what a digital signature is

Space for learners:

4.3 SECURITY ISSUES

The technique which helps to give protection and security to the database against some threats (like accidental or intentional threats) is known as database security. Security problems will extend beyond the data in an organization's database: a breach of security may impact other system components, affecting the database structure in the end. Consequently, database security includes software parts, hardware parts, human resources, and data.

Database security refers to the set of tools, procedures, and mechanisms used to ensure the confidentiality, integrity, and availability of a database.

Appropriate controls, which are distinct in a particular mission and purpose for the system, are required to use security efficiently.

Database security can be considered regarding the following situations:

- Loss of data privacy.
- Loss of data integrity.
- Loss of availability of data.
- Loss of confidentiality or secrecy.
- Theft and fraudulent.

The organization should focus on reducing the threat which can be incurring loss or damage to data inside a database from the listed circumstances above. Because all of the data inside an organization are interrelated, an activity that results in a loss in one area can often result in a loss in another. These scenarios primarily represent areas where the company should concentrate on lowering the risk of data loss or destruction in a database.

Space for learners:

4.4 PRINCIPLES OF DATABASE SECURITY

We need a security model to organize our ideas on security, which may take many different shapes depending on responsibilities, level of detail, and goal. The most important categories are areas of interest (threats, impact, and loss) as well as the actions involved in dealing with them.

The loss of assets is one example of a security risk. Among these are:

- Hardware
- Software
- Data quality
- Data Credibility
- Data
- Availability
- Business benefit

4.5 SECURITY MODELS

The formal description of security policies is called a security model. A security model offers the environment for database considerations, such as implementation and operation, by establishing external criteria for analyzing security issues in general. Security models for specific DBMSs are significantly essential in system design and operation. Security models describe the elements of a database management system (DBMS) that must be deployed to set up and run actual security solutions. Concepts are embodied, regulations are implemented, and servers are provided for such functions. Any deficiencies in the security model will translate either into insecure operations or inefficient systems. For evaluating and comparing security policies, security models are useful tools. We can use security models to check for completeness and consistency in security policies.

The following elements are used to describe security models:

- ❖ **Subjects**: Entities that request access to objects.

- ❖ **Objects**: Entities for which subjects are making the access request.
- ❖ **Access Modes**: various operations performed by the subject on the object (read, write, create, etc.).
- ❖ **Policies**: Enterprise-wide accepted security rules.
- ❖ **Authorizations:** Specification of access modes for each subject on each object.
- ❖ **Administrative Rights**: Who has rights in system administration, and what responsibilities do administrators have.
- ❖ **Axioms**: Basic working assumptions.

Space for learners:

4.6 SOME COMMON THREATS IN DATABASE SECURITY

a. Privilege Elevation: There are some software flaws that attackers can exploit to elevate their access privileges from a regular user to that of an administrator, resulting in misunderstandings of typical analytical data and funds transfers to fake accounts for specific analytical data.

b. SQL or nonSQL Injection: The insertion of arbitrary SQL or nonSQL attack strings into database queries served by web applications or HTTP headers is a database-specific threat. These attacks are vulnerable to organizations that do not follow secure web application coding practices or conduct regular vulnerability testing.

c. Excessive Privilege Abuse: When database users are given various privileges and allowances that go beyond what is required of them, they may be abused for nefarious purposes. For example, if a company user has the ability to change employee residence information, that user could abuse their database update privileges and change someone's salary information.

d. Legitimate Privilege Abuse: This occurs when a legitimate database user abuses their rights to access the database for illegal purposes. This is triggered when a system manager or database administrator abuses their authority and engages in any illegal or unethical behavior.

e. Platform Vulnerabilities: Platform information describes the operating system in use. Vulnerabilities in operating systems such as Windows 2007, Linux, and Windows XP, as well as the additional services installed on a database server, can result in data corruption, illegal access, or a denial of service. A database system's security measures and protection can be overridden by an operating system's flaws.

f. Database Communication Protocol Vulnerabilities: Almost all database retailers' database communication protocols have a significant amount of security flaws. False activities directed at such vulnerabilities can range from unauthorized data access to denial of service and data exploitation, among other things.

g. Denial of service (DoS/DDoS) attacks: In a denial of service (DoS) attack, the attacker floods the target server—in this case, the database server—with so many requests that it can no longer fulfill legitimate requests from real users, and the server becomes unstable or crashes in many cases. The deluge comes from multiple servers in a distributed denial of service (DDoS) attack, making it more difficult to stop the attack.

h. Malware Malware: Malware Malware is software written specifically to exploit vulnerabilities or otherwise cause damage to the database. Malware may arrive via any endpoint device connecting to the database's network.

4.7 CHALLENGES OF DATABASE SECURITY IN DBMS

Given the huge growing number and speed of threats to databases and many other types of information assets, research efforts should focus on data quality, intellectual property rights, and database survival.

1. Data quality –

- To analyze and confirm the quality of data, the database community needs approaches and certain organizational solutions. Straightforward mechanisms such as quality stamps that are displayed on various websites are examples of these approaches. We also require approaches that will enable us to develop more effective integrity semantics verification tools for

Space for learners:

data quality evaluation, based on a variety of approaches such as record linkage.

- Application-level recovery methods are also required to rectify the erroneous data automatically.
- These challenges are now being addressed by ETL (extract, transform, and load) technologies, which are extensively used for putting data into data warehouses.

2. Intellectual property rights –

As Internet and intranet usage grows, legal and informational issues over data are becoming important problems for many organizations. To address these issues, watermarking is employed, which helps to secure content against unlawful copying and dissemination by granting the owner of the verifiable content power.

They are traditionally reliant on the availability of a broad domain within which the objects can be changed while maintaining their fundamental or vital features.

However, further research is desirable to assess the robustness of many of these strategies, as well as to analyze and explore a wide range of approaches or methodologies targeted at preventing intellectual property rights violations.

3. Database Survivability –

Despite disruptive events such as information warfare assaults, database systems must continue to operate and perform their tasks despite decreased capabilities.

A database management system should be able to accomplish the following in addition to making every attempt to avoid and detect attacks:

- **Confident:** We must act quickly to remove the attacker's access to the system and isolate or confine the problem to prevent it from spreading further.
- **Damage assessment:** Determine the scope of the issue, including any failed functions or data corruption.
- **Recover:** To re-establish a normal level of functioning, recover corrupted or loss of data and repair, as well as reinstall, failed functions.
- **Reconfiguration:** Reconfigure the operation to allow it to continue in a degraded condition while recovery occurs.

<u>Space for learners:</u>

- **Fault treatment:** Determine the weakness exploited in the attack to the degree feasible and take actions to avoid a recurrence.

Space for learners:

4.8 CONTROL METHODS OF DATABASE SECURITY

Database security refers to the protection of sensitive data and the prevention of data loss. The Database Administrator is responsible for the database's security (DBA). The following are the primary control mechanisms used to ensure database data security:

1. Authentication
2. Access control
3. Inference control
4. Flow control
5. Database Security applying Statistical Method
6. Encryption
7. RAID Tools
8. Backup and Recovery

These are explained as following below.

1. **Authentication :** Authentication is the practice of determining if a user logs in solely with the permissions granted to him to execute database transactions. A user can only login to the extent of his power, but he cannot access any additional sensitive data. Authentication limits the privilege of accessing sensitive information. These biometric authentication technologies, such as retina and figure prints, can protect the database from unauthorized or fraudulent users.
2. **Access Control :** Unauthorized users' access to the database must be limited by the security mechanism of the database management system. The DBMS manages access control by generating user accounts and controlling the login procedure. As a result, database access to sensitive data is limited to those people (database users) who are permitted to do so, and unauthorized access is prohibited. During the whole login time, the database system must also maintain track of all actions conducted by a certain user.
3. **Inference Control :** This approach is referred to as the statistical database security countermeasures. Its purpose is to

prohibit the user from finishing any inference channel. This approach prevents sensitive information from being disclosed inadvertently. There are two sorts of inferences: identity disclosure and attribute disclosure.

4. **Flow Control :** This prohibits information from reaching unauthorized users. Covert channels are paths for information to pass implicitly in ways that violate a company's privacy policy.

5. **Database Security applying Statistical Method :** The Statistical database security is concerned with protecting personal individual values kept in and utilized for statistical reasons, as well as retrieving value summaries based on categories. They do not allow for the retrieval of personal information.

   This permits access to the database in order to obtain statistical information on the number of employees employed by the company but not to obtain detailed confidential/personal data on a particular individual employee.

6. **Encryption :** This technique is mostly used to secure sensitive information (such as credit card numbers and OTP numbers) and other numbers. Some encoding algorithms are used to encode the data. Unauthorized users will have a pretty hard time decoding this encoded data, whereas authorized users are given decoding keys to decrypt data.

7. **RAID Tools:** RAID (Redundant Array of Independent Disks) is a huge disk array that consists of numerous independent disks that are structured to promote reliability while also increasing performance. Data striping (the data is divided into equal-size partitions) boosts performance by distributing it across many disks in a transparent manner. Using a parity scheme or an error-correcting technique, reliability is improved by storing redundant information across disks.

8. **Backup and Recovery:** Backup is the process of copying the database and log files to offline storage media on a regular basis. A database management system should provide backup capabilities to aid in the recovery of a database in the event of a breakdown. It's usually a good idea to make backup copies of the database and log files on a regular basis and keep them in a

Space for learners:

secure location. The backup copy and the details saved in the log file are used to restore the database to the most recent feasible, consistent state in the event of a failure that renders it useless.

Journaling is the process of storing and maintaining a log file (or journal) of all changes made to the database in order to successfully recover in the case of a failure. A database management system should include logging capabilities, also known as journaling, that keep track of the current state of transactions and database modifications to aid recovery procedures. The benefit of journaling is that in the event of a failure, a backup copy of the database plus the information in the log file can be used to restore the database to its last known consistent state. If no journaling is configured on a failed system, the only way to recover the database is to restore it from the most up-to-date backup version.

STOP TO CONSIDER

A covert channel is any communication channel that can be used by a process to transfer data in a way that goes against the security policy of the system. In a summary, covert channels transmit data using non-standard methods that are incompatible with the system's design.

CHECK YOUR PROGRESS-I

1. Why is database security essential?
2. What is a database threat?
3. What are the control measures for database security?
4. What is RAID in DBMS?
5. What role does backup perform in database security?

4.9 MULTILEVEL SECURITY

The security policy that allows you to classify objects and users using hierarchical security system levels and a non-hierarchical security system is known as Multilevel security.

Multilevel security prevents unauthorized users from accessing information that is classified higher than their authorization level and users from declassifying information.

Space for learners:

The following are the various benefits of multilevel security:

1. Multilevel security enforcement is mandatory and automatic.

2. Methods that are difficult to express through traditional SQL views or queries can be used by multilevel security.

3. To provide row-level security control, multilevel security does not rely on special views or database variables.

4. Multilevel security controls are consistent and integrated throughout the system, allowing you to avoid defining users and authorizations multiple times.

5. Users are unable to declassify information due to multilevel security.

Using multilevel security, you can describe security for Db2 objects and perform other checks, such as row-level security checks. Row-level security checks let you control which users have permission to view, modify, or perform other actions on specific rows of data.

Row access control and multilevel security are mutually exclusive. You can enable column access control on a table with a security label column and enforce it on that column, but you can't do the same with row access control. You can't use row access control to enable a security label column in a table. Vice versa is proper; if a table is activated through row access control, you won't be able to add a security label column to it.

4.10 TYPES OF ACCESS CONTROL

Organizations must decide which access control model to use based on the nature and sensitivity of the data they're processing. Older access approaches include discretionary access control (DAC) and mandatory access control (MAC), but today's most used model is role-based access control (RBAC). Access Control is a permission that allows a user to create or access (that is, read, write, or update) a database object (such as a relation, view, or index), as well as run some DBMS utilities. Offering too many unneeded privileges can compromise security. A privilege should only be granted to a user if that user would be unable to perform his or her activities without it. Following that, the DBMS keeps track of how these privileges are provided to other users and possibly withdrawn, ensuring that only users with the proper privileges have access to an object at all times.

4.10.1 Mandatory access control (MAC)

It is based on system-wide policies that individual users cannot modify. In this technique, each database object is designated a security class, and each user is given clearance for a security class, and users are subjected to restrictions about database object reading and writing. The DBMS decides whether a user may read or write an object based on a set of rules that include the object's security level and the user's clearance. These rules aim to ensure that sensitive information is never passed on to another user without permission. Support for MAC is not included in the SQL standard.

It's worth noting that most commercial DBMSs only provide mechanisms for discretionary access control at the moment. However, government, military, and intelligence applications, as well as numerous industrial and corporate applications, all require multilevel security. Some DBMS providers, such as Oracle, have issued customized versions of their RDBMSs for government usage that include required access control.

Top secret (TS), secret (S), confidential (C), and unclassified (U) are the most common security levels, with TS being the highest and U being the lowest. Other, more sophisticated security classification methods exist, such as those that organize security classes into a lattice. To keep it simple, we will use the system with four security classification levels, where TS ≥ S ≥ C ≥ U, to illustrate our discussion. The Bell-La Padula model, which is widely used for multilevel security, divides each subject (user, account, program) and object (relation, tuple, column, view, operation) into one of four security classifications: TS, S, C, or U. The clearing (classification) of a subject S will be referred to as class(S), and the classification of an object O will be referred to as class(O) .

Based on the subject/object categories, there are two restrictions on data access that has been enforced:

1. S**imple security property**:A subject *S* is not allowed read access to an object *O* unless class(*S*) ≥ class(*O*). This restriction is self-explanatory, and it enforces the obvious rule that no subject can read an object with a higher security classification than the subject's security clearance.

2. S**tar property** (or *-property): A subject *S* is not allowed to write an object *O* unless class(*S*) ≤ class(*O*). This restriction is less intuitive in this instance. It prevents a subject from writing an object with a lower security classification than their own. If this rule is broken, information can flow from higher to lower classifications, which is against multilevel security's basic tenet.

A user (subject) with TS clearance can make a copy of a TS-classified object and then write it back as a new U-classified object, making it visible throughout the system.

STOP TO CONSIDER

The major security method for relational database systems has typically been the discretionary access control approach of giving and revoking privileges on relations. This is an all-or-nothing approach: A privilege is either granted or denied to a user. Many applications require an extra security policy that categorizes data and users according to security classes. This method is referred to as mandatory access control (MAC).

4.10.2 Discretionary access control (DAC)

The granting and revoking privileges is a common method of enforcing discretionary access control in a database system. Consider privileges in the context of a relational database management system. In particular, we'll discuss a privileges system similar to the one created for the SQL language. Many current relational DBMSs use this technique in some form or another. The main concept is to include statements in the query language that allow the DBA and certain users to grant and revoke privileges.

Space for learners:

Types of Discretionary Privileges

The concept of an authorization identifier is used in SQL2 and later versions to refer to, roughly speaking, a user account (or group of user accounts). In place of an authorization identifier, we'll use the terms user and account interchangeably. The database management system (DBMS) must allow selective access to each database relation based on specific accounts. Because operations can be controlled, having an account does not necessarily entitle the account holder to all of the DBMS's functionality. Informally, there are two levels for assigning database system privileges:

a. **The account level.** The DBA specifies the specific privileges that each account has at this level, which are independent of the database relations.

b. The relation (or table) level. The DBA can manage the privileges to access any particular relation or view in the database at this level.

On R, you have access to references. When establishing integrity constraints, the account now has the ability to reference (or refer to) a relation R. This permission can also be restricted to R's specific attributes.

Note that in order to specify the query that corresponds to the view, the account must have the SELECT permission on all relations included in the view definition.

Specifying Privileges through the Use of Views

The views mechanism is a significant discretionary authorization method in its own. If the owner A of a relation R wants another account B to be able to retrieve only particular fields from R, A can construct a view V of R that only includes those characteristics and then grant SELECT on V to B. The same can be said for limiting B to just retrieving particular tuples from R; a view V can be defined by a query that picks only those tuples from R that A wishes B to have access to.

Revoking of Privileges

It may be necessary to temporarily grant a user a privilege in some instances. For instance, the owner of a relation could want to give a user the SELECT privilege for a specific task and then withdraw it once the work is accomplished. As a result, revocation of

Space for learners:

privileges requires a mechanism. REVOKE is a SQL command that can be used to revoke privileges.

Space for learners:

Propagation of Privileges Using the GRANT OPTION

When the owner A of a relation R gives another account B a privilege on R, the privilege might be given to B with or without the GRANT OPTION. If the GRANT OPTION is specified, B has the ability to grant that privilege to other accounts on R. Assume A grants the GRANT OPTION to B, and B subsequently grants the privilege on R to a third account C, which has the GRANT OPTION as well. Privileges on R can be propagated to other accounts in this manner without the knowledge of R's owner. If the owner account A now revokes the privilege assigned to B, the system should automatically revoke all privileges that B spread based on that privilege.

A user's privileges can come from two or more places. For instance, both A2 and A3 may grant A4 a specific UPDATE R privilege. In this situation, even if A2 revokes A4's privilege, A4 will retain it because it was granted by A3. If A3 later revokes A4's privilege, A4's privilege is completely lost. As a result, a database management system that permits permission propagation must keep note of how each privilege was provided so that privilege revocation may be done accurately and completely.

An Example to Illustrate Granting and Revoking of Privileges:

Assume the DBA creates four accounts: A1, A2, A3, and A4, but only A1 is allowed to create base relations. The DBA must use the GRANT command in SQL to accomplish this:

GRANT CREATETAB TO A1;

The CREATETAB privilege is an **account privilege** that allows account A1 to create new database tables (base relations).

To achieved this DBA issue a CREATE SCHEMA command, as follows:

CREATE SCHEMA EXAMPLE AUTHORIZATION A1;

The user account A1 can now create tables in the EXAMPLE schema. Assume that A1 builds the two base relations EMPLOYEE and DEPARTMENT; A1 becomes the owner of these two relations and has ***all relation privileges*** on each of them.

Assume that account A1 wants account A2 to have the privilege to insert and delete tuples in both of these relations. A1, on the other hand, does not want A2 to be able to extend these rights to other accounts. A1 can issue the following command:

GRANT INSERT, DELETE ON EMPLOYEE, DEPARTMENT TO A2;

The owner account A1 of a relation has the GRANT OPTION enabled by default, allowing it to grant privileges on the relation to other accounts. However, because account A2 was not provided the GRANT OPTION in the preceding command, it cannot grant INSERT and DELETE privileges on the EMPLOYEE and DEPARTMENT tables.

Let's say A1 wants account A3 to obtain data from either of the two tables and distribute the SELECT privilege to other accounts. A1 can issue the subsequent command:

GRANT SELECT ON EMPLOYEE, DEPARTMENT TO A3 WITH GRANT OPTION;

The section WITH GRANT OPTION indicates that A3 can now use GRANT to extend the privilege to other accounts. By performing the following command, A3 can provide A4 the SELECT privilege on the EMPLOYEE relation:

GRANT SELECT ON EMPLOYEE TO A4;

Assume A1 decides to revoke A3's SELECT privilege on the EMPLOYEE relation; A1 can then run the following command:

REVOKE SELECT ON EMPLOYEE FROM A3;

The SELECT privilege on EMPLOYEE from A3 and the SELECT privilege on EMPLOYEE from A4 must now be automatically revoked by the DBMS.This is due to the fact that A3 granted that privilege to A4, yet A3 no longer has it.

CHECK YOUR PROGRESS-II

6.What are the main differences between DAC and MAC?

7.What are the drawbacks of discretionary access control?

Space for learners:

4.10.3 Role-Based access control (RBAC)

Space for learners:

RBAC (role-based access control) became popular in the 1990s as a tested method of administering and enforcing security in large-scale enterprise-wide systems. Privileges and other permits are associated with organizational roles rather than individual users, according to the core premise. After that, users are assigned roles that are appropriate for them.

The CREATE ROLE and DESTROY ROLE commands can be used to create and destroy roles. When needed, the GRANT and REVOKE commands can be used to assign and revoke rights from roles as well as individual users. Roles such as sales account manager, purchasing agent, mailroom clerk, department manager, and so on may exist in a company. A number of people can be assigned to each role. The role name receives security privileges that are common to all roles, and anyone assigned to this role acquires those privileges automatically.

RBAC can be used in conjunction with standard discretionary and mandatory access controls to ensure that only approved users who are assigned to specific roles have access to specific data or resources. Users can create sessions during which they can activate a portion of their roles. Each session can have several roles assigned to it, but it only maps to one user or one subject. Many database management systems provide the concept of roles, which can be assigned roles.

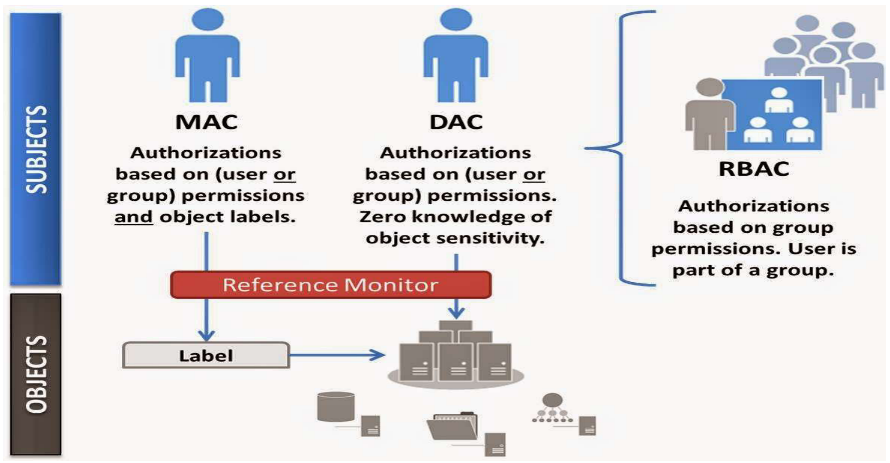

SUBJECTS
OBJECTS
MAC
Authorizations based on (user or group) permissions and object labels.
DAC
Authorizations based on (user or group) permissions. Zero knowledge of object sensitivity.
RBAC
Authorizations based on group permissions. User is part of a group.
Reference Monitor
Label

Fig 4.1: MAC vs. DAC vs... RBAC (**Adapted from:[2]**)

4.11 AUTHENTICATION AND AUTHORIZATION

Space for learners:

As users, we are all aware of most systems' login requirements. In most cases, gaining access to IT resources necessitates a secure login process. This subject is about database management system access, and it provides an overview of the procedure from the standpoint of a DBA. The majority of what follows is on Relational Client-Server Systems.

Other system models change to different extents, but the simple principles remain the same.

Authentication

The client must establish the server's identification, and the server must establish the client's identity. To achieve this, shared secrets (i.e., a password/user-id pair or shared biographic and/or biometrics) are often utilized. It can also be accomplished through the use of a system of higher authority that has already established authentication earlier.

Authentication from a peer system may be appropriate in client-server systems when data (not necessarily the database) is disseminated. It's interesting to note that authentication can be passed from one system to the next. As far as the DBMS is concerned, the result is an authorization identifier. Authentication does not grant any special permissions for certain operations. It only proves that the DBMS believes the user is who he or she claims to be and that the user believes the DBMS is indeed the intended system.

Authentication is a prerequisite for authorization. Authentication is required before authorization may be granted.

Authorization

Authorization refers to a user's ability to carry out specific transactions, such as changing the database's state (write-item transactions) and/or receiving data from the database (read-item transactions).Authorization, which must be done on a transactional basis, is a vector: Authorization (item, auth-id, operation). A vector is a set of data values that are stored at a certain location in the system.

The DBMS functionality determines how this is implemented. On a logical level, the system structure necessitates the use of an authorization server that works in tandem with an auditing server. As the authorization is transmitted from system to system, there is a problem with amplification and server-to-server security. Amplification here refers to the fact that as the number of DBMS servers involved in the transaction grows, so do the security concerns.

Generally, Audit requirements are routinely applied in an ineffective manner. You must log all accesses and all authorization details with transaction identifiers to be safe. There is a requirement to audit on a regular basis and keep an audit trail, which is typically for a long period.

4.12 ENCRYPTION

In certain situations where the DBMS's normal security mechanisms are insufficient, a DBMS can use encryption to protect data. For example, An intruder could steal data tapes or tap a communication line. The DBMS ensures that stolen data is not intelligible to the intruder by storing and transmitting it in encrypted form. As a result, encryption is a technique to provide privacy of data.

4.12.1 Data Encryption

Encryption is the process of transforming plaintext to ciphertext, and decryption is the reverse procedure. The plaintext is the unencrypted message in cryptography. A function with a key parameter transforms the plaintext. The ciphertext is the final result of the encryption procedure. The network is then used to send the ciphertext.

The transmitting end performs encryption, while the receiving end performs decryption. The encryption key is required for the encryption process, and the decryption key is required for the decryption process, as indicated in the diagram. Intruders who do not know the decryption key are unable to convert ciphertext to plaintext. Cryptography is another name for this technique.

The primary idea behind encryption is to utilize a user-specified or DBA-specified encryption key and an encryption algorithm that may

Space for learners:

be accessible to the intruder to encrypt the original data. The encrypted form of the data is the algorithm's output. A decryption algorithm is also available, which takes the encrypted data and the decryption key as input and returns the original data. The decryption algorithm creates garbage without the proper decryption key. The encryption and decryption keys may or may not be the same., but they must have a secret relationship.

There are the following techniques used for the encryption process:

• **Substitution Ciphers**: To mask each letter or group of letters, a substitution cipher replaces them with another letter or group of letters. For instance, A is replaced by D, B by E, C by F, and Z by C. As a result, the attack becomes. Because an intruder can readily predict the substitution characters, substitution ciphers are not very safe.

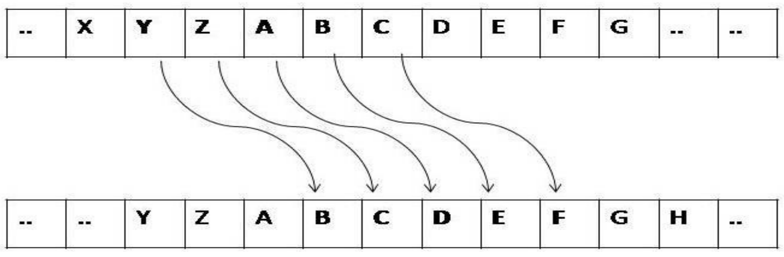
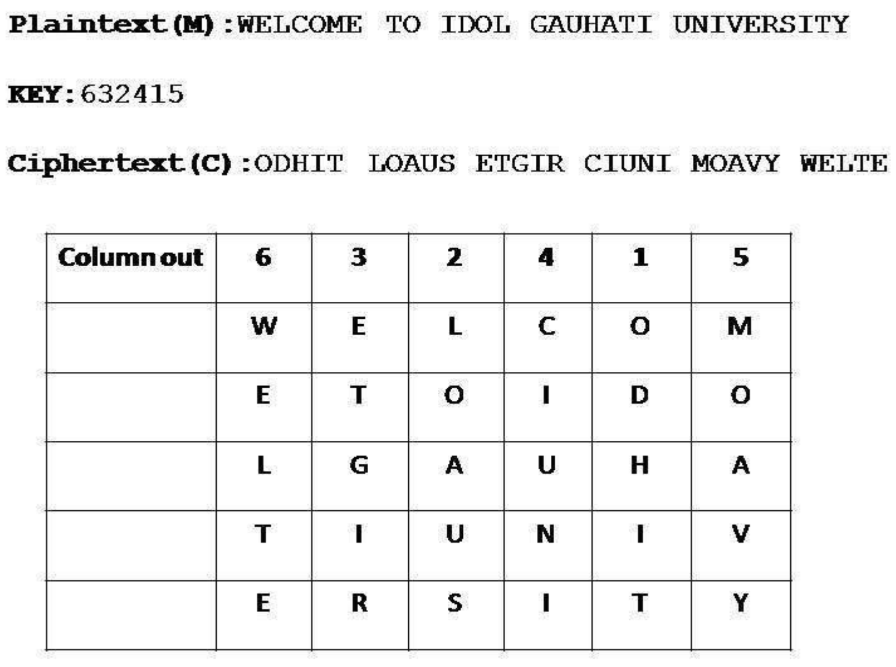


Fig 4.2: Example of Substitution Ciphers

• **Transposition Ciphers**: Substitution ciphers keep the plaintext symbols' order but hide them. The transposition cipher, on the other hand, rearranges the characters without masking them. A key is used in this process. For instance, A may be coded as B. When compared to substitution ciphers, transposition ciphers are more secure.

Space for learners:

Space for learners:



Plaintext(M):WELCOME TO IDOL GAUHATI UNIVERSITY

KEY:632415

Ciphertext(C):ODHIT LOAUS ETGIR CIUNI MOAVY WELTE

Column out	6	3	2	4	1	5
	W	E	L	C	O	M
	E	T	O	I	D	O
	L	G	A	U	H	A
	T	I	U	N	I	V
	E	R	S	I	T	Y

Fig 4.3:Example of Transposition Ciphers

4.12.2 Algorithms for Encryption Process

For the encryption process, there are several widely used algorithms. The following are examples of these

a. **Data Encryption Standard (DES):** On the basis of an encryption key, it performs both character substitution and order rearrangement. The main flaw in this approach is that the encryption key must be communicated to authorized users, and the mechanism for doing so is vulnerable to clever intruders.

b. **Public Key Encryption:**

In recent years, a method of encryption known as public-key encryption has grown in popularity. Rivest, Shamir, and Adheman proposed the RSA encryption scheme, which is a well-known example of public-key encryption. Every authorized user has a public encryption key that is known to everyone, as well as a private decryption key (which is used by the decryption algorithm) that is chosen by the user and is only known to him or her. The encryption and decryption algorithms are assumed to be known to the general public

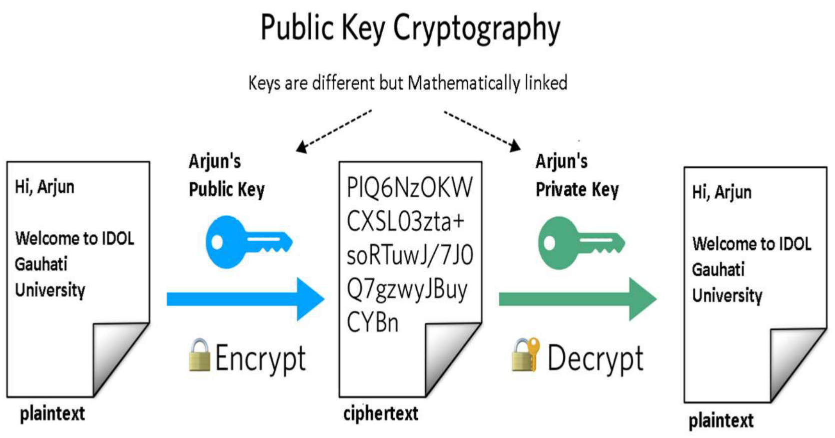

Public Key Cryptography
Keys are different but Mathematically linked
Hi, Arjun
Welcome to IDOL Gauhati University
plaintext
Arjun's Public Key
Encrypt
PIQ6NzOKW CXSL03zta+ soRTuwJ/7J0 Q7gzwyJBuy CYBn
ciphertext
Arjun's Private Key
Decrypt
Hi, Arjun
Welcome to IDOL Gauhati University
plaintext

Fig 4.4: Public Key Encryption process

The following are the most important characteristics of a public key encryption scheme: −

- For encryption and decryption, different keys are used. This is a feature that distinguishes this scheme from symmetric encryption schemes.
- Each receiver has his own decryption key, which is commonly referred to as his private key.
- The receiver must publish his public key, which is an encryption key.
- To avoid spoofing by an adversary as the receiver, some assurance of the authenticity of a public key is required in this scheme. This type of cryptosystem typically involves a trusted third party that certifies that a specific public key belongs to a single person or entity.
- The encryption algorithm is complicated enough that an attacker will be unable to deduce the plaintext from the ciphertext and the encryption (public) key.
- Despite the fact that private and public keys are mathematically related, calculating the private key from the public key is not possible. In fact, designing a relationship between two keys is an intelligent part of any public-key cryptosystem.

Space for learners:

4.12.3 Disadvantages of encryption

Space for learners:

There are the following challenges of Encryption:

1. The management of keys (i.e., keeping them hidden) is a problem. The decryption key must be kept secret even in public-key encryption.

2. Even in an encrypted system, data must be processed in plaintext regularly. As a result, transaction programs may still have access to sensitive data.

3. At the level of physical storage organization, encrypting data causes severe technical issues. Indexing data that is stored in encrypted form, for example, can be extremely difficult.

STOP TO CONSIDER

The sender and receiver of a message in a **symmetric cryptography** system share a single, common key that is used to encrypt and decrypt the message. This is a simple system to set up, and both the sender and the receiver can encrypt and decrypt messages.

Asymmetric cryptography, also known as public-key cryptography, is a system in which the sender and receiver of a message use a pair of cryptographic keys to encrypt and decrypt the message – a public key and a private key. This is a complicated system in which the sender can encrypt the message with his key but cannot decrypt it. The receiver, on the other hand, can decrypt but not encrypt the message using his key.

4.13 DIGITAL SIGNATURE

In e-commerce applications, a Digital Signature (DS) is an authentication technique based on public-key cryptography. It assigns an individual a distinct mark within the body of his message. This allows others to authenticate that message senders are genuine.

To protect against fraud and theft, a user's digital signature is typically different from message to message. The method is as follows −

1. The sender takes a message and calculates the message digest before signing it with a private key.
2. After that, the sender appends the signed digest to the plaintext message.
3. The communication channel is used to send the message.
4. The receiver eliminates the appended signed digest and verifies it with the public key associated with it.
5. The receiver then applies the same message digest algorithm to the plaintext message.
6. If the results of steps 4 and 5 matches, the receiver can be confident that the message is genuine.

Space for learners:

CHECK YOUR PROGRESS-III

8. Is authentication required for authorization?

9.What is plaintext and ciphertext?

10. What do you mean by public key encryption?

Multiple Choice Questions:

11. What is true about data security?

    a. Data security is the protection of programs and data in computers and communication systems against unauthorized access
    b. It refers to the right of individuals or organizations to deny or restrict the collection and use of information
    c. Data security requires system managers to reduce unauthorized access to the systems by building physical arrangements and software checks.
    d. All of the above

12. which statement is used to revoke an authorization,
    a. Alter
    b. Modify
    c. Revoke
    d. All of these

4.14 SUMMING UP

Space for learners:

- The technique which helps to give protection and security to the database against some threats (like accidental or intentional threats) is known as database security. Database security refers to the set of tools, procedures, and mechanisms used to ensure the confidentiality, integrity, and availability of a database.
- The process of ascertaining whether someone or something is who or what it claims to be is known as authentication.
- The practice of granting someone permission to do or have something is known as authorization.
- Encryption is the process of encoding data with a special algorithm that makes it unreadable by any program that doesn't have the decryption key.
- In a computing environment, access control is a security technique that regulates who or what can view or use resources. It is a basic security concept that reduces the risk to the company or organization.
- Discretionary access control (DAC) is a type of security access control that enables or restricts object access based on an access policy set by the owner group and/or subjects of the object.
- Mandatory access control (MAC) is a system-enforced access control method that enforces security policy through clearances and labels.
- A digital signature could be used to confirm that the data was sent by the intended recipient. It is made up of two pieces of data: a string of bits computed from the data being signed using signature algorithms and the private key or password of the person wishing to sign the document.
- Ciphertext is plaintext that has been encrypted using an encryption algorithm. Ciphertext cannot be read until it has been decrypted (converted to plaintext).

4.15 ANSWERS TO CHECK YOUR PROGRESS

Space for learners:

1. It's important to protect the data that any company collects and manages. Database security can protect your database from being hacked, which can result in financial loss, reputational damage, consumer distrust, brand erosion, and non-compliance with government and industry regulations.

2. A database threat is an object, person, or other entity that poses a risk of sensitive data loss or corruption to an asset.

3. These various security controls aid in the management of security protocol circumvention.

i. System hardening and monitoring.
ii. DBMS configuration
iii. Access
iv. Database auditing
v. Authentication
vi. Encryption
vii. Backups

4. Redundant Array of Independent Disks (RAID) is a technology that combines multiple small, low-cost disc drives into an array of disk drives that outperforms a single large, expensive drive (SLED). Redundant Array of Inexpensive Disks is another name for RAID.

5. In data management, making backups of collected data is critical. Human error, hardware failure, virus attacks, power outages, and natural disasters are all protected by backups. If these failures occur, backups can help save time and money.

6. The main difference between DAC and MAC is that DAC is an access control method in which the resource owner determines access, whereas MAC is an access control method in which access is granted based on the user's clearance level.

7. DAC is simple to use and understand, but it does have some drawbacks, such as:

- Inherent vulnerabilities (Trojan horse)
- Grant and revoke permissions maintenance.
- ACL maintenance or capability.
- Limited negative authorization power.

8. Authentication is a prerequisite to authorization because it allows for the secure validation of the subject's identity. After the authentication process is completed, authorization policies begin.

What data you can access is determined by the authorization process.

9. The input to an encryption algorithm is plaintext. The unreadable output of an encryption algorithm is known as ciphertext.

10. The term "public-key encryption" refers to a type of encryption that implements two keys. There are two types of keys: a public key that everyone knows and a private key that only you know.

11. a

12. c

4.16 POSSIBLE QUESTIONS

Short answer type questions:

1. What do you mean by database security?
2. What is the need for database security?
3. What is the principle of database security?
4. Mention the threat of database security in DBMS.
5. What are various control methods used to secure DBMS?
6. Define RAID technology.
7. Define flow control.
8. What is SQL injection?
9. What is the access control method?
10. Describe Authentication and Authorization.
11. Define ciphertext and plaintext.
12. Why is backup essential in DBMS?
13. What is the Mandatory access control method?
14. Differentiate between MAC and DAC.
15. Write a short note on Role-Based Access Control?
16. What is Data Encryption Standard(DES)?
17. What is a digital signature?
18. What do Database Encryption and Decryption mean?

Space for learners:

Long answer type questions:

1. Explain the needs of database security
2. Describe the different threats to database security.
3. Discuss various challenges in database security.
4. Describe the access control method.
5. Explain multilevel security in DBMS.
6. Explain various types of access control methods.
7. Describe the Mandatory access control methods in DBMS.
8. Describe the Discretionary access control method with an example.
9. Explain the Role-Based Access Control (RBAC) method.
10. Describe the data encryption process.
11. Explain how digital signature works.

Space for learners:

4.17 REFERENCES AND SUGGESTED READINGS

[1] Chapter 12. Database Security.
ttps://www.cs.uct.ac.za/mit_notes/database/pdfs/chp12.pdf

[2] Cloud Audit Controls: MAC vs. DAC vs. RBAC.
http://www.cloudauditcontrols.com/2014/09/mac-vs-dac-vs-rbac.html.

[3] Discretionary Access Control Based on Granting and
https://www.brainkart.com/article/Discretionary-Access-Control-Based-on-Granting-and-Revoking-Privileges_11580/

[4] Elamasri R . and Navathe, S., Fundamentals of Database Systems (3 rd Edition), Pearson Education, 2000.

[5] Database Management Systems, Raghurama Krishnan, Johannes Gehrke, TATA McGraw Hill 3rd Edition.