27-8-24

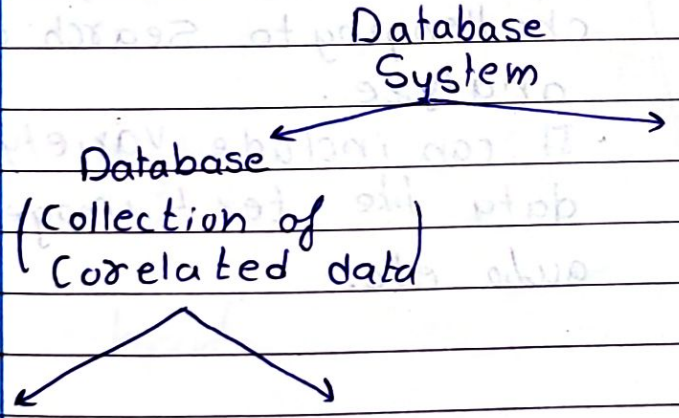
DBMS

Page :
Date :

* Data : Data is statically raw and unprocessed information.
ex. name, class, marks etc.

* Database : A database is a collection of data that is organized, which is also called structured data.

* Database Management System : A database management System is a Software System that is designed to manage and organize data in a Structured manner.
It allows users to create, modify and query a database, as well as manage the security and access control for that database.



* Structured Database :
It is designed to store structured data, which is highly organized and easily accessible.
It uses Schema that defines tables, rows and columns.

ex: Relational databases.

* Unstructured database:
- It is designed to handle unstructured data, which doesn't follow a predefined data model or schema.
- These databases are often more flexible, allowing for the storage of various types of data like text, images, videos, and documents.

ex. NoSQL database: MongoDB, Cassandra, Couchbase.

Structured Data	Unstructured data
• It is highly organized and easily searchable.	• It lacks predefined format or organization, making it more challenging to search and analyze.
• It often stored in tabular format like databases, Spreadsheets.	• It can include variety of data like text, images, video audio etc.

* Files Vs DBMS.

Files	DBMS
• Simple, often unstructure or semistructure	• Highly Structured (RDBMS) or Semi Structured (NoSQL)
• Organization is manual	• Predefined schemas or flexible modes.
• Accessed using file System	• Accessed using query languages or API

Camlin

Page :
Date :

Integrity	Limited data integrity. Integrity must be manually ensured.	Enforces data integrity with constraints, ACID properties.
Concurrency	Difficult to manage; requires manual handling.	Handles concurrent-access efficiently through locking & transactions (Multiuser)
Retrieval	Involves manual searching or basic operations.	Advanced querying, indexing and search capabilities.
Scalability	Limited Scalability,	Highly Scalable
Speed	Slow	Fast.
Memory Utilization	High	Consise memory Utilization
Ease of Use	hard	easy.
Security	relies on file System permissions	Advanced Security.
Redundancy	Higher data redundancy. duplicate data is common.	Controls redundancy through normalization and centralized data management
Performance	Performance degrade	Optimized

Camlin

Page :
Date :

* Purpose of DBMS

The purpose of a database management System is to provide efficient, reliable and convenient way to store, manage and retrieve data.

Primary purposes are

- Data Management
- Data Retrieval
- Data Integrity.
- Data Security.
- Data Redundancy.
- Data Concurrency.
- Backup and Recovery.
- Data Sharing.
- Data Abstraction.

* Types of DBMS.

1. Relational (RDBMS).
   - Data is stored into tables (relations) as rows & columns.
   - SQL is used for querying & managing.
   - ex. MySQL, PostgreSQL, Oracle Database, SQL Server.

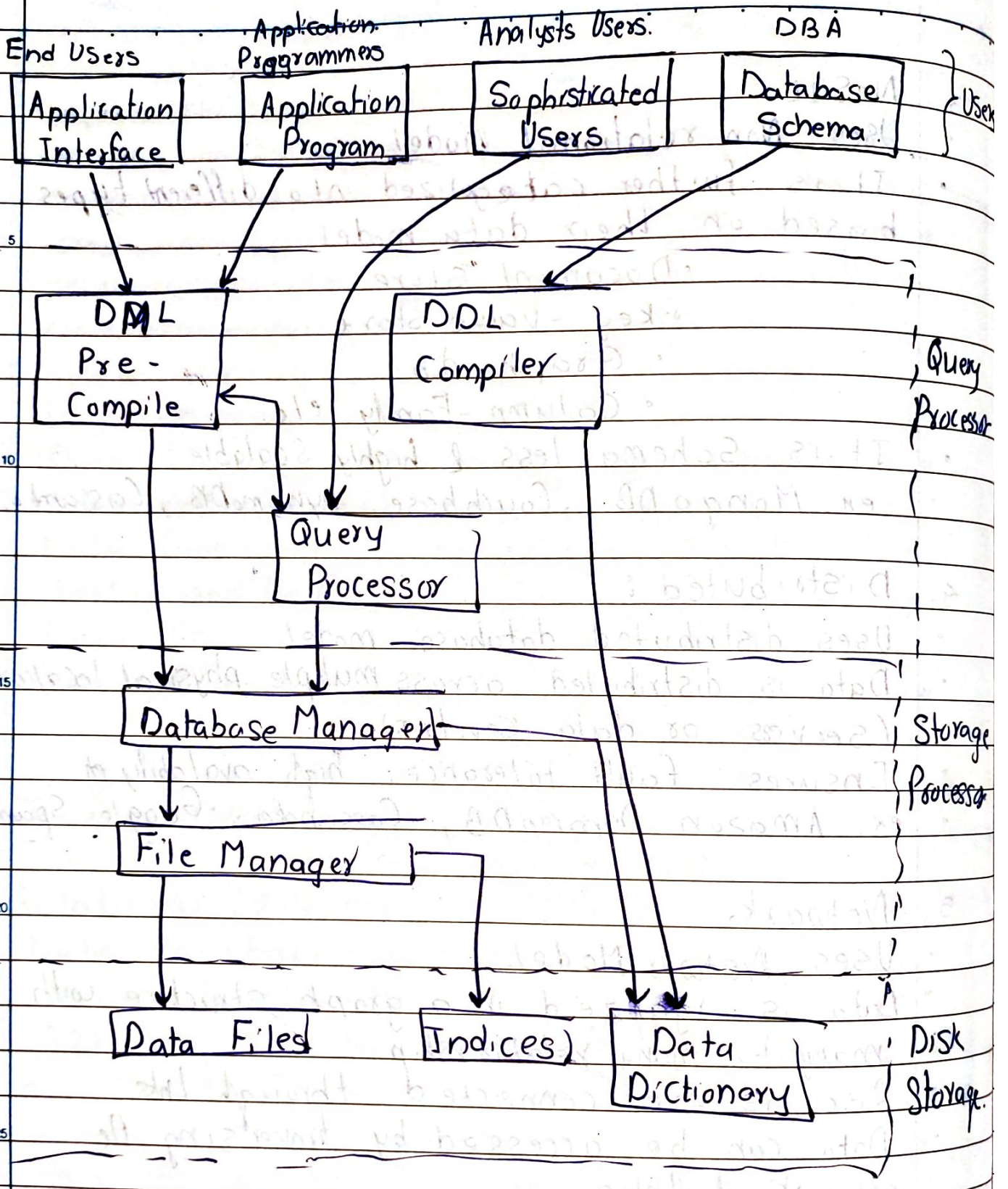
2. Heirarchical DBMS.
   - Uses heirarchical model.
   - Data is organized in a tree-like structure with parent-child relationships (one to many).
   - It is accessed by tree traversal.
   - ex. IBM information Management System.

3 NoSQL
- Uses Non relational model.
- It is further categorized into different types based on their data model.
  - Document store
  - key - value store
  - Graph db.
  - Column - Family etc.
- It is Schema less & highly Scalable.
- ex. MongoDB, Couchbase, DynamoDB, Cassandra,

4. Distributed :
- Uses distributed database model.
- Data is distributed across multiple physical locations (Servers or data Centres).
- Ensures fault tolerance, high availability etc.
- ex. Amazon DynamoDB, Cassandra, Google Spanner.

5 Network.
- Uses Network Model.
- Data is organized in a graph structure with many to many relationship.
- Records are connected through links.
- Data can be accessed by traversing the network of links.
- ex. IDMS (Integrated Data Mem Sy
  IDS (Intr Data Store)

DBMS Architecture / Structure / Components.



* Three Parts that make up the database System are:
  - Query Processor
  - Storage Manager
  - Disk Storage.

Page :
Date :

1. Query Processor Unit :

- The query processing is handled by the query processor.
- It executes the user's query by transforming user requests into instructions that computer can understand.
- Components of Query Processor

i) DDL Compiler (DDL Interpreter).

- Data definition Language is what DDL stands for
- It is used to interpret statements like those used in schema definitions (such as create, drop etc).

ii) DML Compiler :

- Compiler for DML Data Manipulation language.
- It converts DML statements like Select, update and delete into low level instructions.
- It identifies DML commands and converts them into Standard function calls.

iii) Query Processor.

- It is the main component responsible for interpreting and executing SQL queries.
- It takes user queries and pre compiled DML queries. and ensures that they are Syntactically and Semantically correct.

Date :

2 Storage Processor Unit:

- It acts as intermediary layer between queries made and data kept in the database.
- It is incharge for retrevring, storing, updating and removing data from database.
- Components of Storage Processor

i) Database Manager:

- It is responsible for overall management of database, including transaction processing, concurrency control and recovery from failures.

ii) File Manager:

- It is responsible for managing the file where the database's data is physically stored.
- It handles low level storage details.

3. Disk Storage Unit:

- A DBMS can use various kinds of data structures as a part of system implementation in the form of disk storage.
- Components of disk Storage.

i) Data Dictionary:

It contains meta data (data of data)

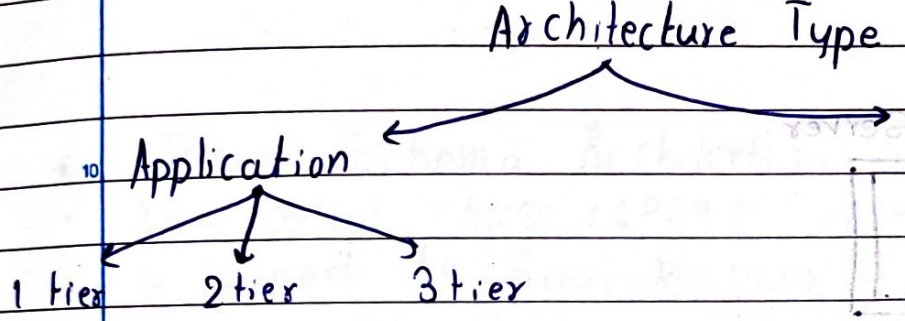
Each object of database has some info about its struction.

ii) Data Files:

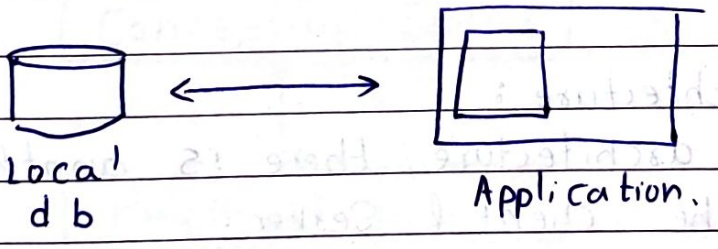
It stores data in files

Page :
Date :

iii) Indices:
These indices are used to access and retrieve the data in a fast and efficient way.



1 Tier : In 1 tier Architecture the database is directly available to the user, the user can directly sit on the DBMS and use it. The server, client database is all present in same machine.



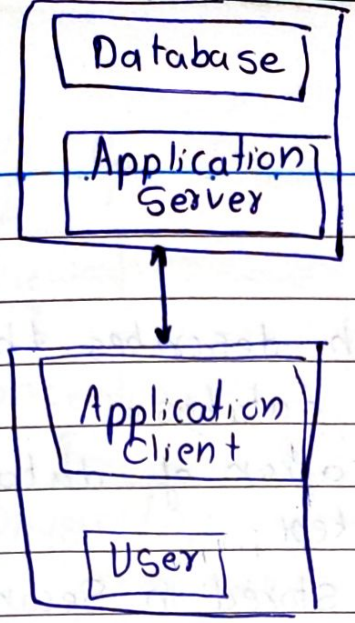
- This is not ideally used
- Adv
  - → Cost effective
  - → Simple Architecture
  - → Easy to Implement.

2 Tier : It is similar to basic client server model. The application at the client end direcly communicates with the database on the server side.

API's like ODBC JDBC are used.
The Server Side is responsible for query processing

Adv.
- Easy maintainance.
- Easy to access
- Low cost
- Scalable.

- 3-trier Architecture :
- In 3 tier architecture, there is another layer between the client & Server.
- The Client does not directly communicate with the Server. Instead, it interacts with an application Server which further communicates to database system.
- It is used in large web applications.

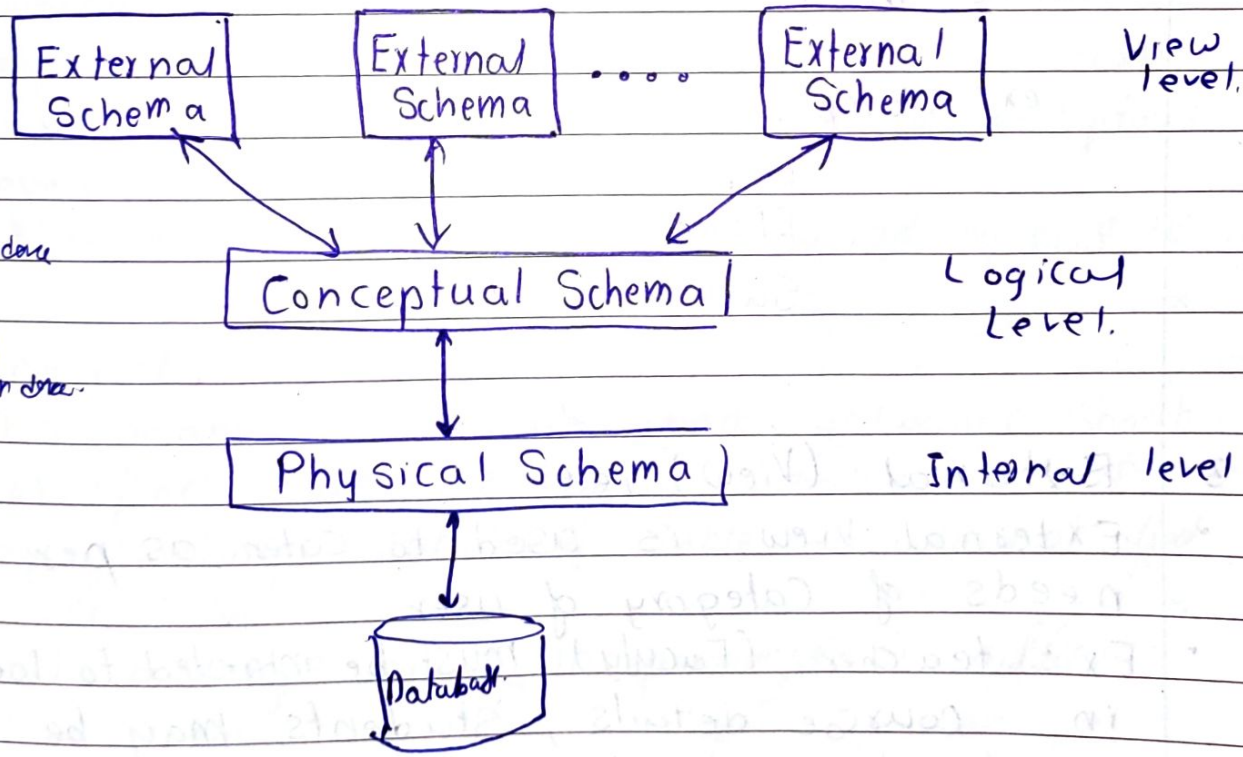


Server.

Client

★ Three Schema Architecture.

- Also called ANSI / SPARC architecture.
- It is used to Seperate user application & actual physical database.
- It contains 3 levels.



- The main objective of 3 level architecture is to enable multiple users to access the same data with a personalize view while storing the underlying data. only once.

Date :

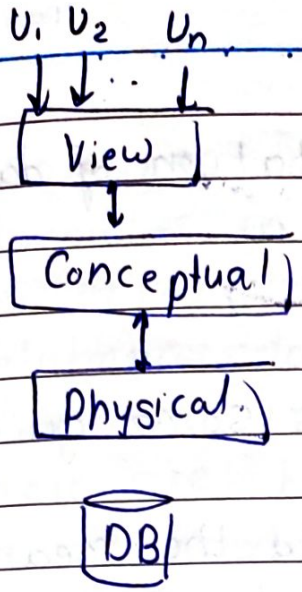
1. Internal Level :
- It has internal schema which describes the physical storage structure of the database.
- The information about location of database objects in the database is kept.
- It states how data is being stored in Secondary memory.

2. Conceptual (Logical) Level.
- It represents the design of a database
- It is represented in the form of various database tables.
- It describes what kind of data is to be Stored in data base.

ex. Employee

Emp no : Integer(4) key
EName : String(15.
Salary : Varchar(8).

3 External (View) Level
- External View is used to cater as per needs of Category of user
- Ex: teacher (Faculty) must be interested to look in course details, students may be interested to look at academic details etc
- Its main focus is data abstraction.



* Data Independence

Data independence means a change of data at one level should not affect another level.

Two types of data independence are present.

* Physical data independence:

- Any change in the physical location of tables and indexes should not affect the conceptual level or external view level.
- It is easy to achieve and implemented by most DBMS.

* Logical independence.

- This means conceptual Schema change should not affect view external schema.
  eg deleting attribute of table should not affect user view.
- But this type of independence is difficult to achieve

Page :
Date :

Schema : It is logical representation of database

* 5 Relational Database Design :
Relational database design has some guidelines to construct.

1. Semantics of attribute : It refers to the meaning of each attribute.
2. 10 Reducing Redundant Information from tuple : Duplicate values should not be present in multiple tuples.
3. Reducing null values in attributes : The value of attribute should not be frequently null.
4. Disallowing possibility of generating Spurious tuples.
15 Refers to avoiding extranious tuples.
(Do not have relations that contain matching attributes other than PK/FK.

ex. Students

S_Id	S_Name	S_Ph

Projects

P_id	P_Name	S_Id	S_Name

Unnecessary

If we join these tables we get two colums S_Name. So projects is S has Spurious tuples.

Page :
Date :

Anomalies in Relationa l Model:

Anomalies in relational model refer to inconsistencies or errors that can arise when working with relational databases, Specifically in the context of data insertion, deletion, and Updation.

There are three categories of anomalies
- Insertion
- Deletion
- Updation

Database anomalies are faults in database caused due to poor management of storing everything in the flat database.
It can be removed with the process of Normalization, which generally splits the database which results in reducing the anomalies in the database.

1. Insertion Anomaly
Inserting wrong values in wrong attributes or inseting a tuple in referencing relation and referencing attribute value is not present in referenced attribute

Ex.

Name	Dept	Age
ABC	CSE	21
pqr	CSE	22
RST	IT	(1)
(Mech)	II	20

2. Deletion and Updation Anomaly:
If a tuple is deleted or update from referenced relation and the referenced attribute value is used by referencing attribute in referencing attribute in referencing relation, it will not allow deleting the tuple from referenced relation.

Name	Dept Id	Dept
ABC	1	CSE
pqr	2	IT

Update →

Name	DeptID	Dept
ABC	1	CSE
pqr	1	IT

To avoid this the following can be used.
- On Delete /UPDATE Set NULL.
- On Delete/UPDATE CASCADE.

* Functional Dependency.
- It is result of interrelationship between attributes or tuples in any relation.
- It helps in normalization and reduce the redundency.
- In relation R, X and Y are two Subsets of Set of attributes. Then Y is said to be functional dependent on x if a given value of X uniquely determines the value of Y

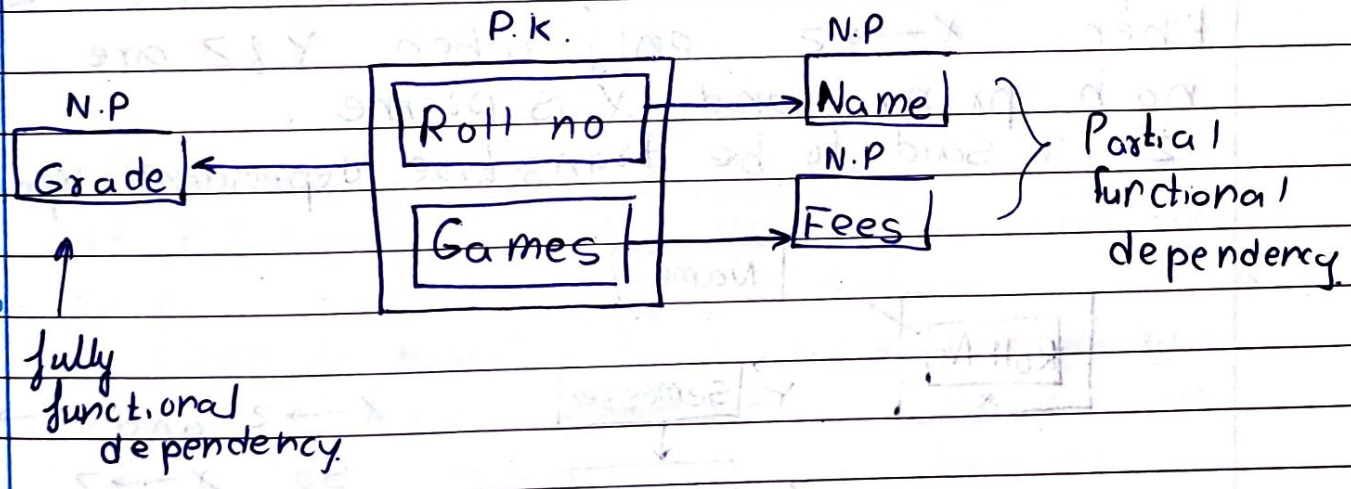
determinant. $X \longrightarrow Y$ dependent

X determines Y
Y is determined by X.

Page :
Date :

* Fully Functional:
let A be a non prime attribute and value of A is completely dependent on prime key attribute then it is said to be fully functional dependency.

* Partial Functional Dependencies:
Suppose you have more than one prime key attribute.
If non prime key attribute does not depend upon all prime key attributes then it is called partial dependency



* Trivial Functional dependency:
If $A \longrightarrow B$ and B is a subset of A then this dependency is called trivial functional dependency.

If $AB \longrightarrow B$
$B \subseteq A$
$\Rightarrow A \longrightarrow B$

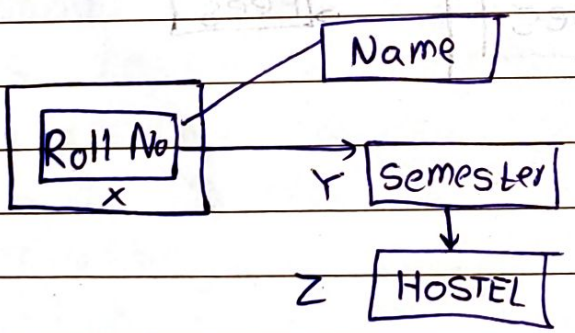
If $A \longrightarrow A$ and $B \longrightarrow B$
and $AB \longrightarrow B$
then $A \longrightarrow B$
by trivial functional dependency

Date :

* Non Trivial Functional Dependency.
If $A \longrightarrow B$ but B is not a subset of A, then such a dependency is called Non Trivial Functional dependency.

If $AB \longrightarrow B$
then
$A \longrightarrow B$ is not allowed in NTFD

* Transitive :
It is due to the dependency between non prime key attributes.
Suppos in relation R, $X \longrightarrow Y$ and $Y \longrightarrow Z$ then, $X \longrightarrow Z$ only when Y & Z are non prime and X is prime.
Z is said to be transitive dependent upon X.



$X \longrightarrow Z$ and $Y \longrightarrow Z$
So $X \longrightarrow Z$.

* Non Transitive Dependency.
Any dependency which is not transitive functional dependent is non transitive dependency.

$X \longrightarrow Y$
$Y \longrightarrow Z$ } Non Trivial.
$X \longrightarrow A$

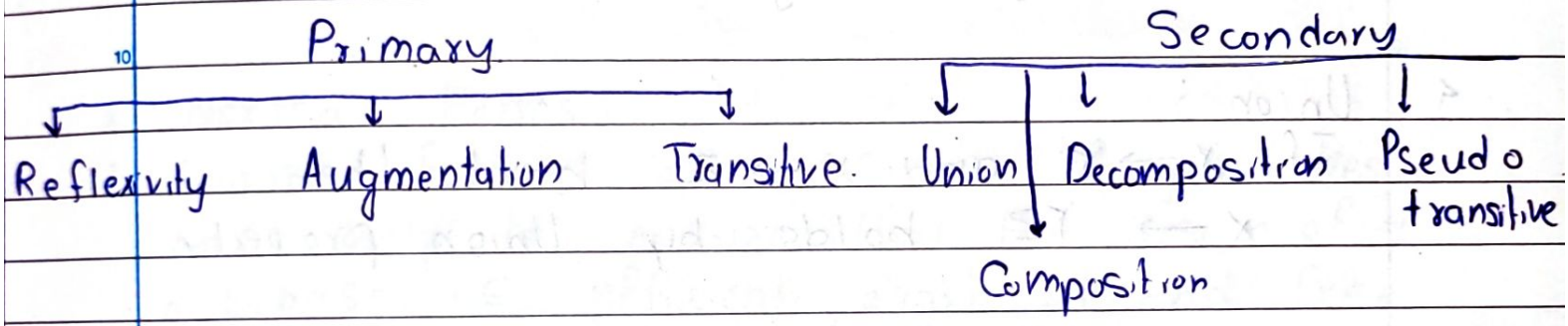
Camlin

Page :
Date :

* Armstrong Axioms.

These are also called as rules or inference of functional dependency.

They are mainly classified into two
- Primary
- Secondary



1. Reflexivity:
If $Y \subseteq X$ then $X \rightarrow Y$ holds by reflexivity rule
ex. {Roll_no, Name} ⟶ {Name} is valid by reflexivity rule.

2. Augmentation.
If $X \rightarrow Y$ is valid then $XZ \rightarrow YZ$ is also valid by augmentation rule.

ex. {Roll_no, Name} ⟶ {Name}

⇒ {Roll_no, Name, Age} ⟶ {Name, Age}.

Date :

3. Transitivity:
If $X \rightarrow Y$ and $Y \rightarrow Z$ both are valid then $X \rightarrow Z$ is also valid by transitive rule.

If {roll_no} → {name}
{name} → {age}

⇒ {roll_no} → {age}

4 Union :
If $X \rightarrow Y$ and $X \rightarrow Z$ hold then
$X \rightarrow YZ$ holds by Union property.

5 Decomposition
If $X \rightarrow YZ$ holds then
$X \rightarrow Y$ and $X \rightarrow Z$ both hold by decomposition.

6 Pseudo Transitive
If $X \rightarrow Y$ & $YZ \rightarrow W$ is valid then.

$XZ \rightarrow W$ is also valid.

7 Composition
If $X \rightarrow Y$ & $A \rightarrow B$ are valid then
$XA \rightarrow YB$ holds.

Page :
Date :

* Normalization :

- It is the process of minimizing redundancy from a relation or set of relations.
- Redundancy in relation may cause insertion, deletion and update anomalies. So, it helps to minimize the redundancy in relations.
- Normal forms are used to eliminate or reduce redundancy in database tables.

* Normal Forms :

- Normal forms are a series of guidelines that help to ensure that the design of a database is efficient, organized and free from data anomalies.
- There are several levels of normalization, each with its own set of guidelines, known as normal forms.

1. First Normal Form (1NF) :

- A relation will be in 1NF if it contains atomic values.
- It states that an attribute of cannot be composite or multivalued.

ID	Course
1	C, C++
2	Java
3	DBMS, C

→

ID	Course
1	C
1	C++
2	Java
3	DBMS
3	C

Date :

2) Second Normal Form (2NF).

- A relation is in 2NF if it already is in 1NF and it does not contain any partial dependency.
- It states that all non prime attributes should be fully functional dependent on candidate key.
- left hand side should be proper subset of candidate key and RHS should be non prime group.

2NF → 1NF + No partial Dependency.

Ex.

Stud_No	Cource_No	Course_fee
1	C_1	1000
1	C_2	1500
2	C_3	2000
3	C_4	1000
3	C_1	1000

This table should be split into two to reduce redundency

Stud_No	Cource_No
1	C_1
1	C_2
2	C_3
3	C_4
3	C_1

Cource_No	Course fee
C_1	1000
C_2	1500
C_3	2000
C_4	1000

Page :
Date :

3. Third Normal Form (3NF).

- A relation is said to be in third normal form, if it is already in 2NF and there is no transitive dependency for nonprime attributes.

Name	Roll_No	Course	Fees.
Ram	1	DBMS	5000
Pooja	2	C	10000
Pooja	3	C++	15000
Rohit	4	DBMS	6000

Here Roll_No → Course
Course → Fees
So it has a transitive dependency.
So, we need to decompose the relation.

Name	Roll_No	Course
Ram	1	DBMS
Pooja	2	C
Pooja	3	C++
Rohit	4	DBMS

Course	Fees.
DBMS	5000
C	10000
C++	15000.

LHS must be candidate key or Super key or RHS is a prime attribute.

4. Boyce Codd 3.5 NF (BCNF)

- It is extension to 3NF.
- It is also called as 3.5 NK.
- It is stricter than 3NF.

A relation is in 3.5 NF if it is already in 3NF and LHS is always a Candidate key or Super key.

3NF ensures that each determinant in a table is a Candidate key. In other words, BCNF ensures that each non-key attribute is dependent only on the candidate key.

Stu_ID	Stu_Branch	Stu_Cource	Branch_Number	Stu_Cource_No
101	CSE	DBMS	B_001	201
101	CSE	CN	B_001	202
102	ECE	VLSI	B_003	401
102	ECE	MComm	B_003	402.

We see that

Stu_ID → Stu_Branch.

Stu_Cource → {Branch_Number, Stu_Cource_No}

We need to decompose.

Stu_ID	Stu Branch.	Stu_Cource	Branch_Num	Stu_Cour
101	CSE	DBMS	B001	201
102	ECE	CN	B_001	202
		VLSI	B_003	401
		MComm	B_003	402

* ~~4NF~~ Fourth Normal Form (4NF)

A relation is in 4NF if it already is in BCNF and it contains no trivial multivalued dependency

Person	Mobile	Food_Likes
ABC	9833/9424	Burger/Pizza
PQR	9191	Pizza.

Person →→ mobile
Person →→ food_likes.

Person	mobile
ABC	9833
ABC	9424
PQR.	9191

Person	Food_Like
ABC	Burger
ABC	Pizza
PQR	Pizza.

Cross Product.

If the cross product of decomposition is equal to original table then our decomposition is proper.

Fifth Normal Form (5NF)

A relation is in 5NF if it already is in 4NF and does not consist any further join. i.e It should have no join dependency also joining must be lossless.
In 5NF the relation must be decomposed in as many sub-relations as possible to avoid redundency.

Subject	Faculty	Year
C	Sai	1
C	Ram	1
C++	Sai	2
Java	Shiva	3

Subject	Faculty
C	Sai
C	Ram
C++	Sai
Java	Shiva

Year	Subject
1	C
2	C++
3	Java

Faculty	Year
Sai	1
Sai	2
Ram	1
Shiva	3

Union of any two tables forms original table.

∴ Decomposed tables are in 5NF

* Dependency Preservation Decomposition:

It is the concept in which a relation is decomposed into multiple relations while preserving functional dependencies.

- When decomposing a relation, it is essential to ensure that all the functional dependencies that hold in the original relation are still enforceable in the decomposed relations.

- A Decomposition $D = \{R_1, R_2, R_3 \ldots R_n\}$ on R is dependency preserving wrt set F of Functional dependency if

$$(F_1 \cup F_2 \cup \ldots \cup F_n)^+ = F^+.$$

* lossless and lossy decomposition

A decomposition of relation R into R_1 and R_2 is said to be lossless if joining to decomposed relations (Natural Join) yields the original relation R.

$$R_1 \cup R_2 = R$$
$$R_1 \cap R_2 \neq \phi$$

- Common attribute must be candidate or super key.

* Indexing:

- Indexing is a technique of organizing data in a way that allows for efficient retrieval.
- It is a critical component in database systems, file systems and other storage solutions.
- It reduces I/O cost.
- Indexes improve speed of query operations by reducing the amount of data the system needs to scan.
- Indexes increase storage requirement
- It slows down write operations as index must be updated with every insertion updation or deletion.

* Structure of Index.

key	pointer

key contains copy of primary key or candidate key of a table.

Date :

* Advantages
- Improved Query Performance
- Efficient data Acces
- Optimized Data Sorting.
- Consistent performance.

* Disadvantage
- More Storage space required
- Increased maintenance overhead.
- Choosing an index can be difficult.

* B Tree :
- Also known as Balanced Tree
- They are Self balancing
- They can Store multiple keys in single node unlike binary tree.
- So, they are also called large key trees.
- All leaf nodes are at same level.
- It is used in multilevel indexing.
- It guarantee time complexity for insertion deletion is O(logn)

* B+ tree.
- It is a variation of the B-tree.
- data pointers are Stored only in leaf nodes.
- All leaf nodes are connected by linked list.
- Internal nodes only has key and block pointers
- Duplicate key are present.

Date :

B+ Tree	B* Tree.
• Separate leaf nodes for data storage and internal nodes fo indexis.	• Nodes store both key and data values.
• leaf nodes form a linked list for efficient range based queries.	• Do not form linked list.
• Higher Order (More keys)	• Lower Order (fewer keys.
• Allows key duplication	• Does not allow key dup!
• Better disk access due to Sequential reads in linked list	• More disk I/O due to non Sequential.
• Better performan	• Balanced perform
• Requires more memory	• Requires high memo
• Grows width wise	• Grows hight wise.
• Random & Sequertion	• Sequenti.
If Order = n.	
max Children = n	
max key = n-1.	
(root) min key = 1.	
(root) m.n (h.ldren = 2	
m.n Children = n/2.	
m.n key (m/2 - 1	
• LHS is Small	• LHS is Small
• RHS is Greater or equal	RHS is greater ~~or equal~~
• All keys present in leaf	• Not all keys present

Page :
Date :

* Transaction Processing

- Transaction is a set of operations used to perform logical unit of work. It generally represent change in database
- It is used to execute a series of operations as a single unit of work.
- These operations must adhere to the ACID properties to ensure data integrity and consistency.

Ex.

Work = transfer money from A to b (transfer 500).

A = 1000
B = 2000 } 500.

R (A)
A = A - 500
W (A)
R (B)
B = B + 500
W (B)
Commit.

After Commit
A = 500
B = 1500

Page :
Date :

* Operations of Transaction:
Following are the main operations of transaction.
- Read (X): Read operation is used to read the value of X from the database and stores it in a buffer in main memory.
- Write (X): Write operation is used to write the value back to the database from the buffer.

ex. R(X);
X = X - 500;
W(X)

- The first operation reads X's value from database and stores it in a buffer.
- The second operation will decrease the value of X by 500 in the buffer.
- The third operation will write the buffer's value to the database

- commit: It is used to save the work done permanently
- Rollback: It is used to undo the work done.

* ACID Properties (Transaction property)
The transaction has four properties. These are used to maintain consistency in a database, before and after the transaction.

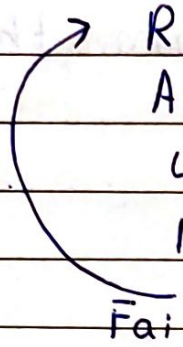
* Property of Transaction
1. Atomicity
2. Consistency
3. Isolation
4. Durability

Camlin

1. Atomicity:

- It states that either all or none are successful.
- A failed transaction cannot be resumed but it is restarted.

ex.



- There is no midway, the transaction cannot occur partially. Each transaction is treated as one unit either it run to completion or is not executed at all.

- Atomicity involves two operations.
  - Abort: If a transaction aboots then all the changes made are not visible
  - Commit: If a transaction commits then all the changes made are visible.

2. Consistency.

- The integrity constraints are maintained so that the database is consistent before and after the transaction.
- Before transaction starts and after the transaction is comple the sum of amount must be same.

A = 1000 → 500
B = 2000 ↙

Sum of A & B before transaction = 3000

A = 500
B = 2500

Sum of A & B after transaction = 3000

- It states that execution of any transaction will leave a database in either its prior stable state or new stable state.

3. Isolation :
- It shows that the data which is used at the time of execution of a transaction cannot be used by second transaction until the first one is complete.
- It is the conversion of parallel transaction into serial scheduling.
- Here conflict and view serializability is used.
- Isolation of transaction prevents dirty read, ...

4. Durability :
- It states that the changes in database should be permanent
- This property ensures that once the transaction has completed execution, the updates and modifications to the datase are stored in and written to disk and they persist even if a system failure occurs.
- It ensures that once a transaction is commited its changes are permanent

Property	Responsibility for maintaining Properties.
Atomicity	Transaction Manage
Consistency	Application programmer
Isolation	Concurrency Control Manager
Durability	Recovery Manager.

- The ACID properties, in totality, provide a mechanism to ensure the correctness and consistency of a database in a way such that each transaction is a group of operations that acts as a single unit, produces consistent results, acts in isolation from other operations, and updates that it makes are durably stores.

- Overall, ACID properties provide a framework for ensuring data consistency, integrity, and reliability in DBMS.
They ensure that transactions are executed in a reliable and consisted manner, even in the presence of system failures, network issues, or other problems.

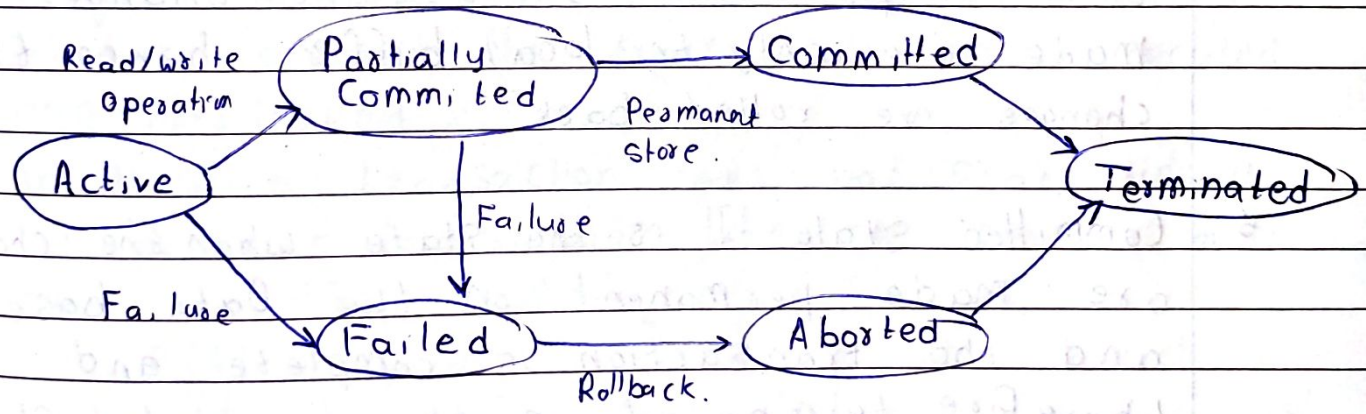
* Advantages of ACID properties in DBMS
- Data Consistency: data remains consistent and accurate after any transaction.
- Data Integrity: ACID properties maintain the integrity of the data by ensuring that any changes to the database are permanent

Date :

- Concurrency Control : It helps to manage multiple transactions occurring concurrently by preventing interference between them.
- Recovery : It ensures that in case of failure or crash, System can recover the dat up to point of failure.

* Disadvantage
- Performace : It can cause performance overhead.
- Scalability : It may cause Scalability issues.
- Complexity : Implementing the ACID properties can increase the complexity.

* States of Transaction :



- Active State : When the instructions of the transaction are running the the transaction is in active state. If all the read /write operations are performed without error then it goes to the partially committed state, if any instruction fails, it goes to the failed state.
- It is the first state of every transaction,
- Partially Committed the transaction is being executed in this State.

Camlin

2 Partially Committed: After completion of all the read and write operation the changes are made in main memory or local buffer. In this state data is still not saved to the database

3 Failed State: When any instruction of the transaction fails, it goes to the failed state or if failure occurs in making a permanent change of data on database.

4. Aborted State: After having any type of failure the transaction goes ~~from~~ failed state to aborted state.
- Since in previous states, the changes made are only to local buffer hence these changes are rolled-back.

5 Committed State: It is the state when the changes are made permanent on the Data base and the transaction is complete and therefore terminated in the terminated State.

6 Terminated State: If there isnt any roll back or the transaction comes from committed state then the system is consistent and ready for new transaction and the old transaction is terminated.

* After aborting the transaction, the database recovery module will select one of two.

Camlin

- Restart transaction
- kill the transaction - go to terminated.

Scheduling

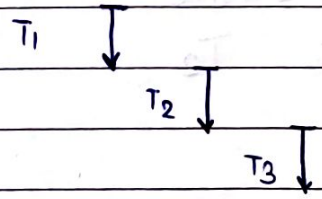
- It is a chronological execution sequence of multiple transactions.
- It is process of lining the transactions and executing them one by one
- When there are multiple transactions the set of operations must not overlap, here is where Scheduling comes in play.

* Schedules are mainly of two type.

1. Serial Schedule:
Schedules in which the transactions are executed non-interleaved.
In this a transaction does not start until its previous transaction does not end.

ex.

T_1	T_2
R (A)	
W (A)	
R (B)	
	W (B)
	R (A)
	R (B)



- It has greater consistency.
- It has greater waiting time.
- Its performance is low.

2 Parallel Schedule.

- In this type of scheduling the operations of multiple transactions are interleaved.
- It can give rise to ~~consistenc~~ concurrency problem also its consistency is low.
- It has lower waiting time
- It has higher performance.
- Transactions are executed in non serial manner, so a transaction does not need to wait completely till its previously transaction ends.
- It is of two types.
  1) Serializable : 2) Non Serializable.

T_1	T_2
read (A)	
A = A - N	
	read (A)
	A = A + M
write(A)	
reade (B)	
	write (A)
B = B + N	
Write(B).	

Page :
Date :

Conflicts in Parallel Scheduling

- Conflicts in parallel scheding occur when multiple transactions interleave their operations on shared data in a way that can compromise database consistency.
- These conflicts arise due to operations like read, write performed simultaneously by different transactions.

Read - Read	→	No Conflict
Read - Write	→	Conflict
Write - Read	→	Conflict
Write - Write	→	Conflict

1. Write Read Conflict (dirty Read Problem):
- One transaction writes a data item, and another transaction reads it before the first transaction commits or roll backs.

ex.

T_1	T_2
R(A)	
A = A - 50	
w(A)	
	R(A)
Failure (Rollback)	

- In dirty read problem one transacton updates an item of the database, and somehow the transaction fails, and before the data gets rollback, the updated database item is accessed by another transaction.

2 Read Write Conflict (Unrepeatable read)

- Also called as inconsistent retrievals problem
- One transaction reads a data item and another transaction modies the same data item before the first transaction finishes.
- It also occurs when two different values are read from same database item.

A=10

T_1	T_2
R (A)	
	R (A)
	A = A - 10
	W (A)
	Commit.
R (A).	

In T_1 first read value 10 and second read value 0 this causes conflict.

3 Write Write Conflict (Irrecoverable / Lost Update)

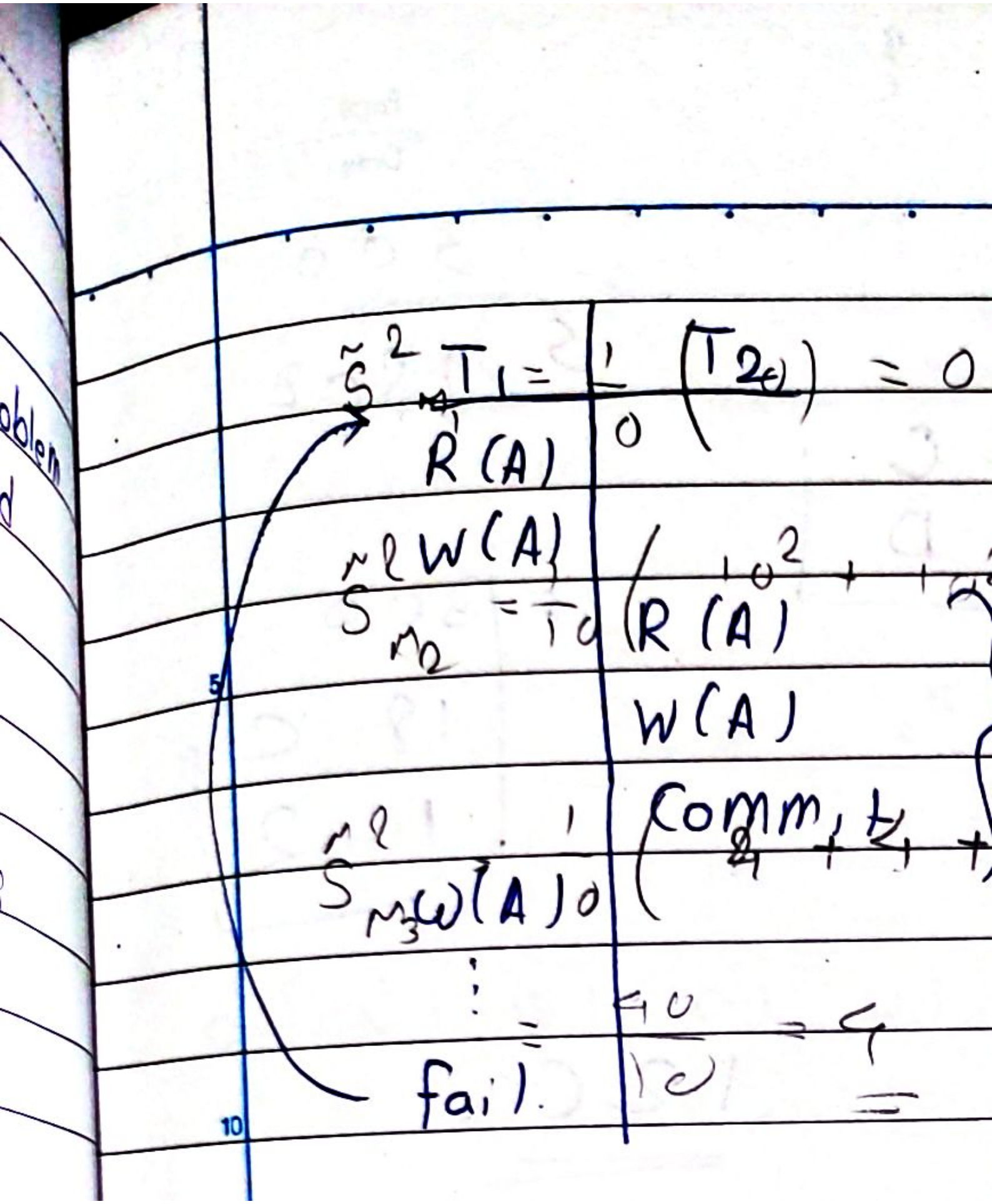
The problem occurs when two different database transactions perform the read/write operations on the same item in an interleaved manner

- It can also happen when two transaction write a data item at same time causing over writing.

ex T_1 : Write (A)

T_2 : Write (A)

if T_2 overwrites T_1 T_2 is lost.


$\tilde{S}^2$ T_1 =
R (A)
W (A)
Commit
fail.