INTERVIEW PREPARATION NOTES !

CS FUNDAMENTALS !

- → DBMS
- → COMPUTER NETWORKS
- → JAVASCRIPT
- → OOPS
- → OPERATING SYSTEMS
- → NODE JS BASICS
- → LINUX COMMANDS
- → GIT COMMANDS
- → MONGODB

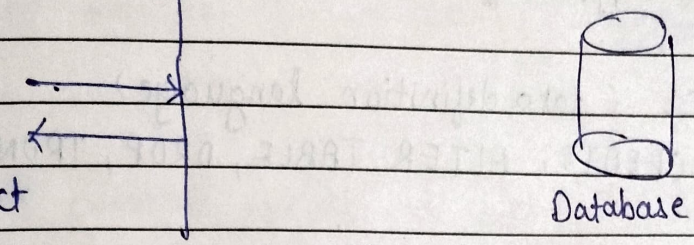
- RUGWED BODHANKAR

SQL (IDBMS) (Structured Query Language) 24/7/23

RDBMS - MySQL, Oracle, etc., IBM.

Page :
Date :

CRUD - (create, read, update, delete)



MySQL - Software used to fetch data using SQL.
(open source RDBMS)

SQL	MySQL
1) Query language (crud operation karpan RDBMS pe)	1) Itself a RDBMS.
2) Way to access data	2) CRUD operation done on it. using SQL.

- For Fetching whole date
  SELECT * FROM is used
- For inserting data to table
  INSERT INTO is used.

* Data types in SQL -

(Fixed sized) CHAR - string (0-255) = VARCHAR (variable datatype) (better in space) जेवढी space पाहिजे तेवढीच देतो.

TEXT - string (0-65535)

BLOB (Binary large object) - string (0-65535)

INT - (-∞ to +∞)

FLOAT -

DOUBLE -

ENUM -

SET -

DATE - YYYY-MM-DD

BOOLEAN - 0/1

BIT

Camlin

TINYINT → (-128 to 127)

UNSIGNED TINYINT → (0 → 255)

Page :
Date :

* SQL Types of command -

1) DDL (data definition language) (DRCAT)
CREATE, ALTER TABLE, DROP, TRUNCATE, RENAME
↳ table ko khali krna

2) DQL (Data query language)
SELECT

3) DML (data modification language) (DINU)
INSERT, UPDATE, DELETE

4) DCL (Data control language)
GRANT, REVOKE

5) TCL (transaction control language)
ROLLBACK, COMMIT, SAVEPOINT

Managing DB
1) CREATE DATABASE IF NOT EXISTS db-name;
2) USE db-name
3) DROP
4) SHOW DATABASES;
5) SHOW TABLES;

* DRL (Data retreival language)

Worker cha naav and salary pahije asel tr.
SELECT FIRST_NAME, SALARY FROM worker;
order of execution Right to Left.

Dual Table are dummy tables created by MySQL, helps users to do certain obrious actions with referring to user defined tables.

SELECT 55+11; (result given in individual guid)
SELECT now(); (time)

Page :
Date :

- WHERE (condition)
SELECT * FROM worker WHERE SALARY > 80000;

- BETWEEN
SELECT * FROM customer WHERE age between 0 AND 100;
↳ inclusive ↲

- IN (reduces OR condition)
SELECT * FROM officers WHERE officer_name in ('LAKSHAY', 'DEEPIKA');

SELECT * FROM worker WHERE DEPARTMENT IN('HR','Admin', 'Account');

- NOT (except that)
S*F worker WHERE DEPARTMENT NOT IN('HR','ADMIN');

- ISNULL (null value wala return karega)

- Pattern searching (wildcard - (%, _)

'%pa%' abcpadb
pabc
adpa

'_pa_' - apab

S*F worker where first_name LIKE '_i%';

- Sorting
S*F worker ORDER BY salary; (Ascending)
salary DESC; (Descending)

- Distinct values -
select DISTINCT department from workers
(gives distinct department).

* Data Grouping - (Aggregation count)

GROUP BY - Aggregation of (COUNT, SUM, AVG, MIN)

Page :
Date :

select department, COUNT(*) from worker group by department.
(also can use COUNT(department))
(should be same.)
(if not applied aggregation fnc it will work as distinct)

select department, AVG(Salary) from worker group by department.

* HAVING - (filtering in GROUP BY)

select department, COUNT(department) from worker group by department HAVING COUNT(department) > 2 ;

WHERE VS HAVING

WHERE is used to filter rows from table based on specific cond^n.
HAVING is used " " from groups " ".

WHERE can be used with SELECT, UPDATE & DELETE keywords while GROUP BY with select.

* DDL - (constraints)

1) Primary key
id INT PRIMARY KEY, or PRIMARY KEY (id);

2) Foreign key - (F.K refer P.K of other table)

FOREIGN KEY (cust_id) references Customer (id).

3) Unique -

4) CHECK - create table account (
:
CONSTRAINT acc_balance_check CHECK (balance > 1000));

5) DEFAULT -

```
create table account (
    balance INT not NULL Default 0;
    );
```

set default value as 0;

an attribute can be Fk and Pk both in table.

6) ALTER operations - (on table)

a) ADD new column

```
ALTER TABLE account ADD interest FLOAT NOT NULL
DEFAULT 0;
```

b) MODIFY

```
ALTER TABLE account MODIFY interest DOUBLE NOT NULL
                                    default 0;
```

c) CHANGE column (change name)

```
alter table account CHANGE COLUMN interest saving_interest
  FLOAT NOT NULL DEFAULT 0;
```

d) DROP COLUMN;

* DML

1) INSERT

```
INSERT INTO Customer ( id, cname )
values (121, 'Bob);
```

2) UPDATE

```
UPDATE Customer SET address = 'Mumbai', Gender = 'M'
            WHERE id = 121
```

UPDATE multiple rows,

```
SET SQL_SAFE_UPDATES = 0;
UPDATE Customer SET pincode = 110000;
```

(by default it is set in SQL that any unauthorised user should not change whole data so we have to run query of SAFE_UPDATES = 0;

Carmlin

d) DELETE FROM customer where id = 121;

Referential Constraint

Page :
Date :

① INSERT ② DELETE Constraint

a) On Delete Cascade

b) on Delete Set NULL;

On Delete Cascade - if entry deleted from parent then uske corresponding all entries will be deleted from child too.

On Delete set NULL - if entry deleted from parent then foreign key table entries won't be deleted but set to NULL in child table.

e) Replace - 1) Data already present, Replace

2) Data not present, INSERT

REPLACE INTO customer (. . .)
values (. . .) ;

(column wise combination) (Faster, difficult)

* JOINS -

Foreign key establish relations.
using relations how I fetch data of JOINS.

Parent child

Left Table

L_1	L_2	key → P.k
-	-	1
-	-	2
-	-	3
		4
		5
		6

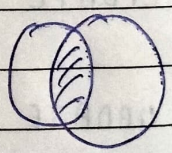
Right Table

F.k ← key	R_1	R_2
1		
2		
3		

Inner Join (return record that have matching values in both tables.)

Resultant table

key	L_1	L_2	R_1	R_2
1				
2				
3				

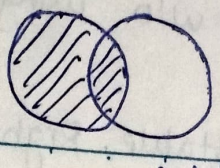


To apply join there should be common attribute.

Syntax.

select C.*, O.* FROM customer AS C INNER JOIN orders AS O. ON C.id = O.cust-id ;

LEFT JOIN



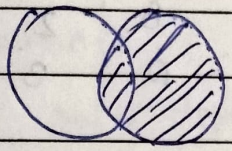
in syntax only right INNER JOIN.

Page :
Date :

Key	L_1	L_2	R_1	R_2
1	—	—	—	—
2	—	—	—	—
3	—	—	—	—
4	—	—	NULL	NULL
5	—	—	"	"
6	—	—	"	"

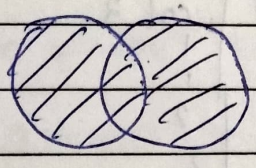
Returns all record of left table, and the matched records from right Table.

RIGHT JOIN



vice-versa of LEFT JOIN

FULL JOIN



(LEFT JOIN U RIGHT JOIN)

```
select * from lefttable as l LEFT JOIN Righttable as r
    on l.key = right.key.
```

(UNION)

" " RIGHT JOIN " "

CROSS JOIN (Cartesian product)

L_1	L_2
1	A
2	B

R_1	R_2
3	C
4	D

Resultant :

L_1	L_2	R_1	R_2
1	A	3	C
1	A	4	D
2	B	3	C
2	B	4	D

SELF Join - (emulate 'INNER JOIN' Alias 'As'

```
select e1.id , e2.id, e2.name FROM employee as e1
  INNER JOIN employee on e2 ON e1.id = e2.id.
```

Q. can we use join w/o using join keyword?

→ Yes.

Page :
Date :

select * from lefttable, Righttable WHERE
lefttalue.id = Righttalue.id;

* SET operations - (Row wise combination) (Datatypes of columns from each table should be same)

Table 1

col1	col2
A	1
B	1
C	2

Table 2

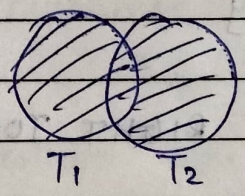
col1	col2
A	1
B	2
D	3

Union $T_1 \cup T_2$

col1	col2
A	1
B	1
C	2
B	2
D	3

select * from table1 UNION select * from table2.

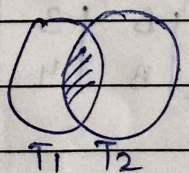
Set operations combine two or more select statements.



if we apply full JOIN above

col1	col2	col1	col2
A	1	A	1
B	1	B	2
C	2	NULL	NULL
NULL	NULL	D	3

INTERSECT -



select * from T_1
INTERSECT
select * from T_2;

X → emulate

select DISTINCT id from T_1
INNER JOIN T_2 using (id);

(A = 1)

$T_1 - T_2$

Syntax

Page :
Date :

select id from T_1 LEFT JOIN T_2 using(id) WHERE T_2.id is NULL ; (B-1 and C-2)

* SubQueries - (alternative to Joins) (easy, slower)

$q_1(q_2)$
↙ ↳ inner.
outer

outer depends on inner query.

select * from table where id IN(select id from table where name = 'Lakshay');

exists mainly in 3 clauses -
1) WHERE 2) SELECT

WHERE - select * from Employee where age > (select avg(age) from Employee);

FROM - select max(age) from (select * from Employee where fname like '%a%');

* Corelated SubQuery - (inner query refers outer query)
(Inner query is driven by outer query)

find 3rd oldest employee.

~~select * from~~ employee e_1 WHERE
3 = (SELECT COUNT(e_2.age) FROM
Employee e_2 WHERE e_2.age >= e_1.age);

32
44
22
31
21

// for each e_1.age inner → completely one time.

1) e_1.age = 32
Linner - 32 >= (2)
2) e_1.age = 44 (1)
3) e_1.age = 22 (4)
4) 31 = (3) = ans.

* SQL views -

UNION - column wise
JOIN - row wise merge

Page :
Date :

CREATE VIEW custom_view AS select fname, age from Employee;

select * from custom_view;

(जितना चाहिये उतना ही दिखायेगा).

- SQL query to find N^th Highest Salary.

→ select salary from employee order by salary desc limit(n-1,1)

IDBMS (I/B) 40 Questions

is a set of applications that enable users to create and maintain a database. DBMS provides tool for performing various operations such as inserting, Deleting etc.

helps to overcome problems like data inconsistency, data redundancy etc.

e.g. XML, Window Registry

RDBMS stores data in form of tables, more effiecient.

e.g. MySQL, Oracle DB etc.

- A database is an organized, consistent and logical collection of data that can be easily be updated, accessed & managed.
- DBMS extracts data from database in form of queries given by the user.
- Accessing data is harder in traditional files due to unorganized nature.
- integrity check, data isolation, atomicity, security are issued with traditional file systems.

4) Adv of DBMS

→ 1) Data sharing 2) Data independence 3) Data security
4) Backup and recovery facility.

A Atomicity — C consistency — I Isolation — D Durability.

Page :
Date :

This property reflects the concept of either executing the whole query or executing nothing at all.

This reflects that the data remains consistent before and after transaction.

This ensures that each transaction is occuring independently of the others.

This ensures that the data is not lost in cases of a system failure or restart and is present in same state as it was before.

Note - NULL value in database is different as that of blank space or zero

- The process of collecting, extracting, transforming and loading data from multiple sources and storing them in one database is data warehousing. It is acts as central repository.

Q. Diff. levels of Data abstraction - (Hiding irrelevant data from use

→ 1) Physical level - lowest level, consists of data storage descriptions and details are hidden from system admins, developers and users.

2) logical level - level on which developers and system admin works and it determines what data is stored in database and what is relationship between data points.

3) View level - database, hides details of table schema from users. (View level data abstraction - (e.g. Result of Query)

Q. E-R model (Entity relationship)

→ is a diagrammatic approach to a database design where real world objects are entities and relationships between them are mentioned.

Entity - R-W-O having characteristics.
e.g. Student, Teacher.

Entity Type - characteristics which uniquely identifies the entity. Student by student-id.

Entity set - set of all entities present in specific database.

Q. Relationship among tables in DBMS.

Row (X) ~~Column~~ Row (Y) // Type

particular row → link → singular row — one-to-one

single row — many rows — one-to-many

multiple — multiple — many-to-many

Q. Intention - database schema used to define description of database. remains unchanged.
Extension - measure of number of Tuples in DB. (Snapshot of DB) and its value keep changing when tuples are created, destroyed etc.

Q. Delete command → works on where clause and deletes only row. (slow)
Truncate → without where and deletes whole table. (fast)

Q. Shared lock - required for reading data. Multiple transactions are allowed to read the data items in shared lock.
Exclusive - write operation. no multiple trans, prevents inconsistency in DB.

Camlin

Normalization is process of reducing Redundancy by organizing the data into multiple tables. (reduce redundancy in Table)

Page:
Date:

Denormalization it combines tables which have been normalized into single table so data retreival becomes faster. (JOIN)

* Normal Form -

1st NF - a) No multivalued attribute, only single valued.
2) Atomic value.

2nd NF - a) Should be in 1st NF and no partial dependency only full dependency.

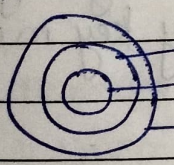
↳ (AB) → C
candidate key

A → C
B → C

3rd NF - a) In 2nd NF and No transitive Dependency.
X → Y → Z (Here T.D)
(when a non-prime attribute depends on other non-prime attributes)

BCNF - a) In 3rd NF + L.H.S must be C_K or S_K

Q. Different keys. C_K and P_K C S_K



unique keys are Primary key with NULL values.

Foreign key defines an attribute that can only take the values of ~~two or more columns~~ present in one table common to the attribute.

Q. 2 tier and 3 tier architecture -
→ 2 tier - client server architecture

client →
client → application → Database
client →

eg. IRCTC etc.

3-tier.. more seaure and accessible.

client
client → GUI → application → Database
client

SQL vs MySQL-

SQL is language used for retreiving and manipulating structured databases. MySQL is relational database m. syste like SQL server that is used to manage SQL databases.

- Foreign key comprises of collection of fields in a table that essentially refers to the primary key in another table.

Database - Group of Interrelated Records.

* Keys in SQL - (combination of attribute which uniquely identifies record)
1) Super key - (सबसेज्यादा combination)
2) candidate key - (min. subset of super key)
3) Primary key - (only 1 in table)
4) Alternate key - Candidate key other than primary key.
5) Foreign key - record that refers to primary key in another table.

* Indexing - Think of indexing like a book's table of contents that helps you quickly find the pages about specific topics.

DBMS can use the index to narrow down the search, making data retreival faster. A database index doesn't store the entire rows but provides pointers to the actual data.

* Sharding - horizontally partitioning a large database into smaller. Each shard contains a subset of data and can be hosted on seperate servers.

Sharding is used to distribute the load and improve performance and scalability of a database system.

Page :
Date :

- Partitioning is a technique used to divide stored database objects into seperate servers. performance ↑, can manage huge chunks of data optimally.

Partitioning

Horizontal
(Independent chunks of data tuples are stored in different servers)

↓

Sharding technique to implement it.

Pros - 1) Scalability
2) Availibility.

Cons - 1) complexity, making partitioning mapping.
2) Not suited for analytical type of queries, as data is spread across different DB instances.

Vertical
(Need to access diff servers to get complete tuples)

(Due to delay in response in 1 single DB partitioning is required)

Adv - 1) Parallelism
2) Availibity
3) Performance
4) Manageability.
5) Reduce cost.

* Clustering is the process of combining more than one servers or instances connecting a single database. One server may not be enough to manage the amount of data or no. of requests. that is when cluster is needed.

* Replication is maintaining multiple copies of the same data on different servers or database instances. used to improve availability or fault tolerance

Note - Clustering optimizes data storage for query performance, while replication ensures data availability and fault tolerance

Computer Networks - 05/08/23

Page :
Date :

Types of Networks.

1) LAN - local area (home, office, factory)
2) WAN - Wide (Geographical area, country or continent)
3) MAN - over a city
4) GAN - Global
5) PAN - Personal (Bluetooth devices)

- VPN -

VPN is a private WAN built on Internet. It creates a tunnel between different networks using internet. By using VPN, a client can connect to the organizations network remotely.

Adv -
a) Cheaper compared to WAN
b) Secure transactions and confidential data transfer b/w offices located in diff. geographical locations.
c) Disguises the online identity.

Types - 1) Access VPN
↙
provides connectivity to remote mobile users and telecommuters. (lost cost)

2) Site-to-Site VPN
↳ large companies having branches in diff locations.

Intranet Extranet

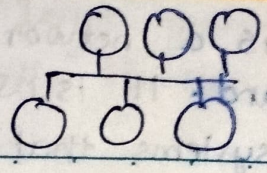
Node - Any communicating device in a network is called Node. N.
e.g. laptop, printers.

Link - refers to the connectivity between two nodes.

Network Topology - It is the physical layout of network, connecting different nodes using the links.

Camlin

Bus Topology -

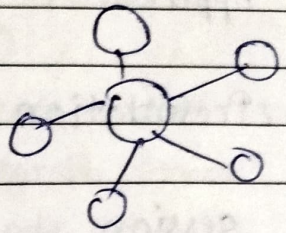


a) All nodes are connected using central links known as the bus.

b) useful for smaller no. of devices.

c) If main cable damaged, it will damage whole network.

Star Topology -



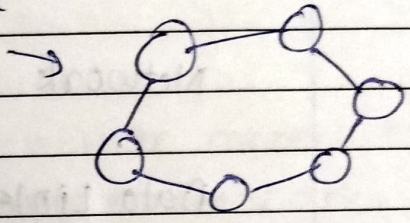
a) connected to central node.

b) Home/offices c) more robust

d) Easy to troubleshoot.

Ring Topology -

a) Expensive & hard to install and manage

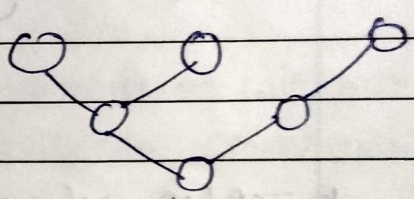


Mesh Topo -

a) Each node → one or many nodes.

b) rarely used

Tree Topo -



a) Bus + star Topology.

b) Star networks connected to single bus.

- IP address is 32 bit dynamic address of node in the network it has 4 octets of 8-bit each with each number with a value upto 255. (A|B|C|D|E)

Private IP address -

A - 10.0.0.0

B - 172.16.0.0

C - 192.168.0.0

open system Interconnection

OSI model - is a network architecture model based on the ISO standard. It is OSI model as it deals with connecting the systems that are open for communication with other systems.

Date :

7 layers -

- Application → contains variety of protocols that are commonly needed by users
- Presentation → concerned with syntax and semantics of info. transmitted.
- Session → allows users on diff. machines to establish session btw them.

(Application, Presentation, Session — Software layers)

- Transport → Heart of OSI → Accept data from above layer, split it into smaller units and pass it to network layer
- Network → controls op. of subnet
- Data Link → transform a raw transmission facility into line that appears free of undetected transmission errors.
- Physical → concerned with transmitting raw bits over a communication.

(Network, Data Link, Physical — Hardware Layers)

* TCP/IP reference model - compressed version of OSI model with only 4 layers.

High reliability

- Application - HTTP SMTP RTP DNS
- Transport - TCP UDP
- Internet (Network) - IP ICMP
- Data Link - DSL SONET 802.11 Ethernet

(protocols)

same as OSI model.

delivers IP packets to the destination

↓ Physical

contains all higher-level protocols.

HTTP - an application layer protocol build upon the TCP. It uses port 80 by default. Stateless protocol.

HTTPS - secure HTTP. It enables secure transactions by encrypting the communication and also helps identify network server securely. uses 443 by default.

- SMTP - (Simple mail transfer protocol)
  SMTP sets the rule for communication btw servers. This rules helps the software to transmit emails over Internet. It is always - listening mode on port 25.

- Router can only send data to similar networks. operates at the network layer.

- TCP/IP is set of rules that decides how computer connects to the internet and how to transmit data over the network. Uses three way handshake model to establish connection which makes it more reliable.

- UDP based on datagrams. uses simple transmission without any hand-shaking which makes it less reliable. (connectionless) (UDP packets are light weight) (very fast) (DNS-UDP)

- ICMP is internet control message protocol. It is a network layer used for error handling. mainly used by network devices like routers for finding the network issues and crucial for error reporting. uses port 7 by default.

- DHCP (Dynamic Host configuration protocol) is an application layer protocol used to auto-configure devices on IP networks enabling them to use TCP and UDP-based protocols. Port 67. It helps to get the subnet mask, IP address and helps to resolve the DNS.

- ARP (Address resolution Protocol) - (IPV4 to MAC addresses) Network-level used to convert logical address (IP address to device's physical address)

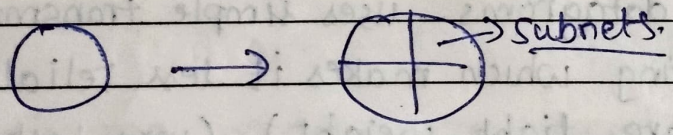
- FTP - application layer proto used to transfer files and data reliably and efficiently btw hosts. 27 port.

- MAC address is media access control address. It is 48-bit or 64-bit unique identifier of devices in network. also called physical address + NIC used at data link layer. NIC is hardware component in the networking device using which a device can connect to net.

- MAC address helps to identify the device and IP address helps to identify the device connectivity on network

Q. Subnet.

⇒ It is a network inside a network achieved by the process called subnetting which helps divide a network into subnets enchances security and higher routing efficiency.



Hub	Switch
a) Physical layer	a) Data link layer
b) Half-duplex transmission mode	
c) less complex, less cheaper	c) effective and intelligent
d) less speed upto 100 MBPS	d) high speed in GBPS
e) less efficient	

ip config - command used in microsoft OS to view and configure network interfaces.

ifconfig - MAC, LINUX, UNIX OS - - -

Q. Firewall -

→ security system which is used to monitor the incoming and outgoing traffic and blocks the same based on the firewall security policies. wall btw internet & private network either a hardware device, software program or combination of both. (filter out IP packets based on rules)

Q. Unicasting - message send to single node from source.
Anycasting - . . . any nodes from source. mainly used to get the content from any servers in content delivery system.
Multicasting - subset of nodes from source then it is multicasting. used to send the same data to multiple receivers.
Broadcasting - send to all nodes from source. DHCP and ARP in local network use broadcasting.

- Network layer is responsible for routing.
- DNS uses UDP as the transport layer protocol.

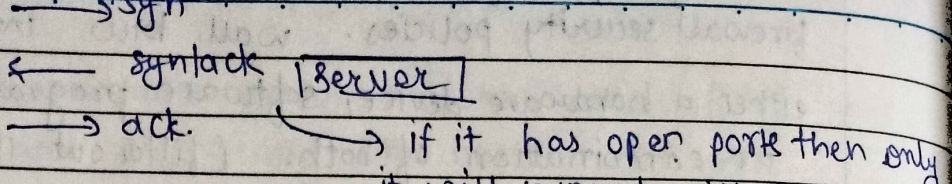
URL - universal resource locator

Q. What happens when we type URL?
DNS (is a database that maintains name of server and IP)

1) first it checks browser cache (maintains a record for websites previously visited.)
2) Then it makes system calls to OS to find record of DNS cache
3) Checks router which has its own cache of DNS records.
4) If everything fails it goes to ISP, then ISP DNS recursive search comes into picture, search will repeatedly continue from DNS server to another DNS server until it either finds IP address or returns error (unable to find)
5) So the domain name which we have entered get connected to IP address and comes to browser.

6) Then browser will build connection with server.

7) Uses TCP 3 way handshaking models for data transfer



8) then browser passes a HTTP request and request will convert it into HTML and we get the data. GET

---

1) Application layer, used by Network Applications such as Chrome, firefox. (uses A.L protocols such as HTTP/HTTPS)

A.L provides services to Network Applications with the help of its protocol (FTP, SMTP, HTTP etc)

2) P.L - (receives data from A.L in character format)
It majorly performs 3 processes.

1) Translation. Data Compression. Encryption

DFGJUYRG → 100011101101 → 10101101 ← 100110011

ASCII → EBCDIC

Lossy
lossless
Decreases the file size in result increases data sharing speed.

SSL
secure socket layers

3) Session layer - (helps in session management, Authentication and Authorization)

4) Transport Layer - (protocols are TCP and UDP)

Full Duplex

(data unit:- TCP segment)

HTTP uses TCP

www
FTP
(Fully transmitted data)

live streaming, video games, music
TFTP, DNS

T.L involved in Segmentation, Flow control, Error Control, Connectionless & connection oriented transmission.

5) Network layer (Gets Data from Transport layers).
Units here are called Packets.

Page :
Date :

functions of Network layer are logical addressing, Path Determination and Routing. IPV4/IPV6

↳ method of moving data packets from src → destination.

6) Data Link Layer - performs 2 major functions.
data unit is called ethernet Frame
1) Access the media (Framing)
2) Controls how data is placed and received from the media (media Access control or Error Detection)
↳ Data encapsulation (Adds header/tailer to IP packet)

7) Physical layer - (converts Binary sequence into signals) → media
(signal generated is determined by the type of media used to connect two devices)
Electrical - LAN cable,
Light - optical fibre
Radio - Air / Vacuum

* HTTP -
1) Port no. 80
2) Not reliable but use TCP to achive it.
3) Stateless (not save requests data)
4) Commands (Head, Get, Post, Put, Delete, Connect)
5) HTTP is Non persistent.

* FTP - → non persistent → persistent.
1) Port 20 (Data) & 21 (Control)
2) Reliable 3) Stateful. 4) Synchronous.

* SMTP - (sync & asynchronous)
1) Port 25

5) Network layer (Gets Data from Transport layers).
Units here are called Packets.

Page :
Date :

functions of Network layer are logical addressing, Path Determination and Routing. (IPV4 / IPV6)
↳ method of moving data packets from src → destination.

6) Data Link Layer - performs 2 major functions.
data unit is called ethernet Frame
1) Access the media (Framing)
2) Controls how data is placed and received from the media (media Access control or Error Detection)
↳ Data encapsulation (Adds header / tailer to IP packet

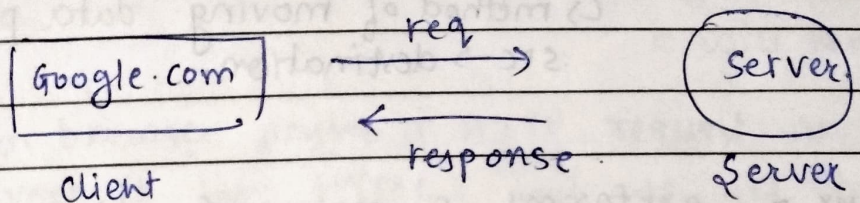
7) Physical layer - (converts Binary sequence into signals) → media
(signal generated is determined by the type of media used to connect two devices)
Electrical - LAN cable,
Light - optical fibre
Radio - Air / Vacuum

* HTTP -
1) Port no. 80
2) Not reliable but use TCP to achive it.
3) Stateless (not save requests data)
4) Commands (Head, Get, Post, Put, Delete, Connect)
5) HTTP 1.0 Non persistent.

* FTP - → Non persistent → persistent.
1) Port 20 (Data) & 21 (Control)
2) Reliable 3) Stateful. 4) Synchronous.

* SMTP - (sync & asynchronous)
1) Port 25

* TLS (Transport Layer Security)
- for providing security in Transport Layer.
- derived from SSL (handshake protocol).
- provides secure connection b/w client & server.
- used by HTTP, SMTP.



0 - 1023 are reserved ports.
HTTP - 80
MongoDB = 27017

- Bridge is present at D.L.L. It is repeater (generate signal over the same network before signal becomes too weak or corrupted.) with add on functionality of filtering content by reading the MAC addresses of source and destination
It is also used to interconnecting 2 LAN's working on same protocol.

- SSH- and Telnet allow remote access to servers, SSH provides strong encryption and security, making it suitable for secure communication over untrusted networks.
Telnet is old network protocol and unsafe of being use in public network due to lack of encryption, so has been replaced by SSH for secure remote access.

- Cookie is a unique string and is stored in our browser when visiting any site for first time.

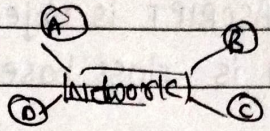
* DNS - (Domain name system) (part of application layer)
.io .org .com

T.L takes care of congestion control (traffic control)
congestion control algo built in TCP

TCP uses Checksums (to check data corrupted or not)

- Transport → Segments.
- Network → Packets.
- Data Link → Frames.

* An IP Address is used to locate a device on a network.
* A MAC address is what identifies the actual device.



- 192.168.2.30
  - 192.168.2 → Network address (subnet ID)
  - 30 → device address / host ID

A will send msg to network of IP address and find the PC with corresponding MAC address.
There are two types of entry in arp
1) dynamic (not permanent, periodically)
2) Static (someone manually enters)

- In Network Layer
  - Routers → Nodes
  - Links → Edges

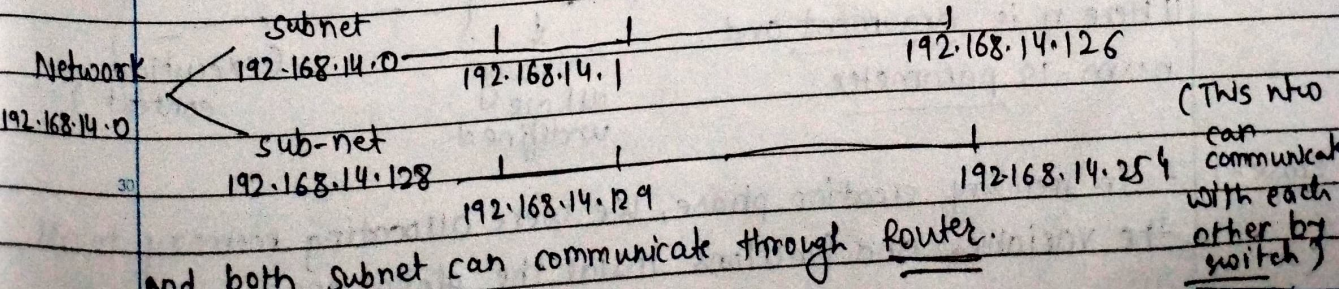
} Created by Control Plane.

- IP (Internet Protocol)
  - IPv4 → 32 bit, 4-words (4 bytes)
  - IPv6 → 128 bits.

Class A - 0.0.0.0 — 127.255.255.255.
B - 128 - 191
C - 192 - 223
D - 224 - 239
E - 240 - 255

Packets - Header is of 20 bytes.
1 byte = 8 bits.

- A subnet is logical subdivision of IP networks.



and both subnet can communicate through Router.

Camlin

JAVASCRIPT

JAVASCRIPT and JAVA are programming Language but they differ in syntax, execution environment and use cases. Both support OOPS concept. ReactJS on other hand is a Javascript library primarily used for frontend developement.

- JAVASCRIPT is object oriented language which is prototype based.
- JAVA is class base object oriented language.

In JAVASCRIPT object are created using constructor functions or object literals, and inheritance is achieved through prototype chaining. whereas in JAVA objects are created using class constructors and inheritance is achieved through class hierarchies.

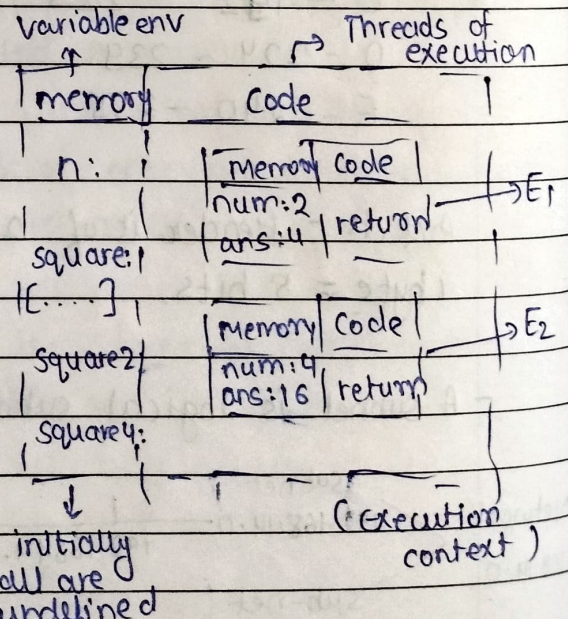
- JAVASCRIPT is single-threaded and uses an event driven, non-blocking I/O model, making it suitable for asynchronous programming. It uses callback functions, promises and async/await for handling asynchronous operations.

- JAVASCRIPT is synchronous single-threaded language. (can execute one command at a time)

e.g.

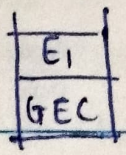
```
var n = 2
function square(num) {
    var ans = num * num;
    return ans; }

var square2 = square(n);
var square4 = square(4);
// Here n is argument and
num is parameter.
```



Phase-1 - In memory creation phase, we were allocating memory to all the variables and functions inside the global space.

call stack.



E1, after executing E1 it will pop and E2 will push into stack.

Page :
Date :

After all execution call stack gets empty.

- In a arrow function like

```
var getName = () => {
    console.log("....");
}
```

getName behaves as a variable and in memory is undefined

```
var getName2 = function () {
}
```

It will still behave as a variable.

- Shortest JS program is a empty file. (It creates window, this also it creates memory. (global object) (this === window)

Note - Anything which is not inside function is global space.

```
var a = 10;
fnc b() {
  var x = 10 }
```

console.log(a) → 10
console.log(x) → not defined (cause it is inside function)

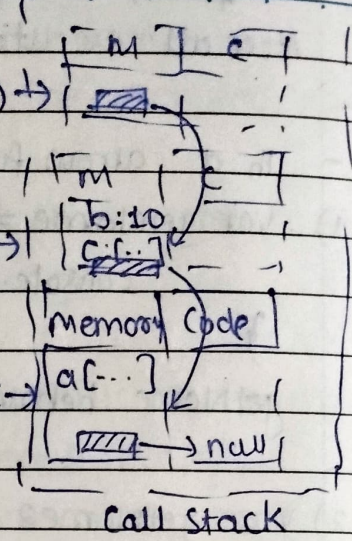
Note - undefined != empty, it takes up its own memory but we can assume it like a placeholder for the time being until the variable is assigned some value.

- scope and lexical environment.
  ↳ where to put code kind of thing.

Page :
Date :

```
function a() {
    var b = 10;
    c();
    function c() {
    }
}

a();
console.log(b);
```



(Here c has access to all the variables)

Here c is lexically present in a and a is lexically present in GEC.

Also it is referencing to its lexically parent (c → a), so if something is not present in c it will check in its lexical parent (a); Whole chain of lexical environment is known as scope chain.

- Lexical env = local memory + reference to lexical env parent.

- console.log(a); → a is hoisted and is undefined

```
let a = 10;
var b = 100;
```

↕ Temporal dead zone (Phase till it is hoisted and gets some value)

and variables cannot be accessed in temporal dead zone.

You cannot access let variables on window object like (window.a) gives undefined.

Block is something which combines multiple statements. also known as compound statement

Page :
Date :

```
if(true){
    var a=10;
    console.log(a);
}
```
→ block.

```
{ var a=10;
  let b=20;
  const c=30; }
```

b and c are hoisted in block scope and a in global and all are set undefined.

b and c cannot be accessible outside block scope whereas a is accessible anywhere.

if a variable is once again var a=100; and we do outside block scope console.log(a). we will get 10 var a=10; shadows the outer value.

In case of let it is not same and prints 100 to output. (same with const)
shadowing also works in functions.

- Closures - (function around its lexical environment) - uses - Module Design Pattern, memoize, setTimeouts, Iterators, Functions like once., maintaining state in async world.

```
- function x(){
    var i=1
  setTimeout(function(){
      console.log(i);
  },3000);

  console.log("Hello");
  x();
```

It prints Hello first and 1 after 3 seconds

1) Closures can be used for data hiding & encapsulation.
2) unused variable are automatically deleted in High level programming by garbage collector. Closures callocate a lot of memory which cannot be deleted so this acts a disadv.

Closure - function + lexical scope (bundled together)

- If there is some function incrementing counter, so it is accessible to other function, for data hiding we can wrap it into other function closure.

```
function counter() {
var count = 0;
return function incrementcounter() {
    count++;
    console.log(count); }
}

var counter1 = counter();
counter1();
```

- this keyword in JS - this stores the current execution context of the JS program.
- The strict context prevent certain actions from being taken and throws more exception.

1) var - function level scope (accessible anywhere)
2) let and const - (block level scope) (allows reassignment)

↳ var only accessible within the block-level in which they are defined.

let and const are used over ~~cons~~ var can help prevent issues like variable hoisting and variable reassignment that fiqh occur at function-level scope.

variable hoisting

- Temporal dead zone, let and const cannot be accessed until they are declared (Reference error) whereas var can be done

- lexical scope - (scope is programs textual structure). with lexical scope, a variable or fnction can access all variables from its parent scope up to global scope but no scope can access variables from the functions defined inside it. determined at compiled time.

e.g. const a = 7 global scope

function b() {
  const c = 3; local scope
  c.l(a);
  c.l(c);
}

function d() {
  const e = 5;
  c.l(c); }

b();
d();

b() → 7, 3
d() → Reference error
c is not defined.

- closures in JS allow a function to access variables from an outer function. A closure is created when a function is bundled together with reference to its surrounding state, known as lexical environment. closures can be used to create private variables and methods.

~~INTRODUCTION~~ - closure example -

```
function add() {
    var counter=0;
    function plus() { counter+=1; }
    plus();
    return counter; }
```

The plus function here is a closure because it has access to counter variable in outer function. closures can have access to all outer function scopes, including block scope and module scope

- Arrow functions - useful for anonymous functions passed as callbacks or event hooks. can be used with await keyword to handle promises and make asynchronous code more readable.

- Promises - are objects that represent the completion of an asynchronous operation and can be used to handle the result of async. actions.

The await keyword can be used inside an async function and makes the fnc wait for a promise to be resolved or rejected before continuing.

Async/await and promises are used to handle asynchronous code in JS. Promises are cleaner way to handle asynchronous code that allows for chaining .then() and .catch() methods, while async/await provides a more procedural syntax that makes asynchronous code look similar to synchronous code with better error handling capabilities.

OOPS

Page :
Date :

Class user-defined datatype which has properties.

```
Class Hero {
    char name[100];
    int health;
    char level;
};
```

} properties

```
int main() {
    // Creation of Object
    Hero h1;          // Hero Ramesh
```

ble - In empty class the and we create an object then we will get 1 byte of memory.

if we want to access the properties we can do it using '.' operator.

```
cout << " health is:" << ramesh.health << endl;
```

If we have to access any data member which is private in your class, you can access it by getter, setter.

↓ Fetch ↓ cond^n

Static memory allocation -
1) Fast access
2) Global variable (used for)
3) Reserving memory for an object during compilation phase.
4) Fixed memory

Dynamic
1) Reducing memory footprin
2) Variable size data structur (arrays, linked lists)
3) Allocating memory for objects at runtime, during execution

Object Creation

Static allocation -

```
Hero a;
a.setHealth(80);
a.getLevel('A');
```

Dynamic Allocation

```
Hero *b = new Hero;
b->setLevel('B');
b->setHealth(70);
```

```
cout<< (*b).level << endl;
        or
cout << b->level << endl;
```

Constructor
- → object creation invoke default constructor
- → No return type
- → No i/p parameter

this stores the address of current object.

constructor taking arguments is parameterized

```
Hero S(70,'C');
S.print();
```

```
Hero R(S);
R.print();
```

// Copy constructor

both will have same values.

```
Hero( Hero &temp ){
    this->health = temp.health;
    this->level = temp.level;
```

Here we pass by reference in place of pass by value to avoid infinite loop.

- Default copy constructor — Shallow Copy (creates a new obj and copies content of original into new)
- In deep copy you duplicates all the data creating a independent copy.
- In shallow copy only duplicates the outer surface (structure) of an object and maintains reference to same nested object.

Shallow copy shares the same reference as other copy created hence change in copy1 will reflect to change in copy2. However in deep copy has its own independent copy of Address object. so change doesn't affect.

Camlin

Destructor - to deallocate memory (same properties as constructor)

Page :
Date :

```
~Hero() {
    cout << "Destructor called" <<
}
```

for static object → Destructor Automatically called.
for dynamic object → Destructor manually called.

```
Hero *b = new Hero();
delete b;    // manually called.
```

- Syntax to access static int (static member)

```
datatype classname :: fieldname = value;    (Initialise outside class)
```

(:: scope resolution operator)

- static functions can only access static members.
  - ↳ obj create krne ki need nahi h.
  - → this keyword - X

* Encapsulation - (wrapping up data members and fnc)
- ↳ Information hiding
- properties
- methods/behaviour

Fully encapsulated class - all properties - private.

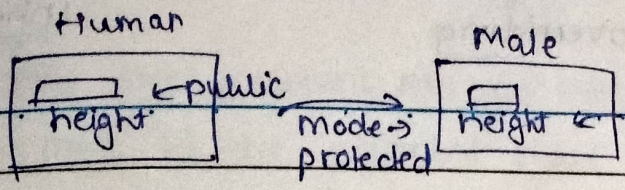
Adv - Data hide → Security ↑

* Inheritance - properties Inherit. (most powerful feature but more time to process)

```
class parent_class { };
class child_class : access modifier parent_class { };
```

(access modifier: public/private.)

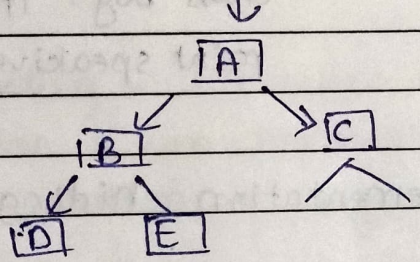
private data member of any class cannot be inherited.



→ can be accessed inside own class only (use getter / setter)

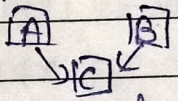
if superclass property is private then it is not accessible in any case.

- Types of Inheritance -
  - ↳ Single → (Animal → Dog)
  - 2) Multi-level → multiple classes (A → B → C → D → E)
  - 3) Multiple → two classes Animal & Human, third class will inherit both
  - 4) hierarchical



parent for more than one class.

In ambiguity we use scope resolution operator.



obj : A :: func();

Adv -
1) Code reusability
2) Hierarchical organization.

* Polymorphism - (existing in multiple forms)
  - many
  - forms

(Polymorphism offers a lot of flexibility in OOPS)

Compile time polymorphism -
→ 1) fnc overloading
2) oprator overloading.

Same name se function chaljayega but input alag hona chahiye. ditto same nakoye.
↳ arguments

* Run-time polymorphism (Dynamic) - (possible through inheritance)
↳ method overriding

Page :
Date :

```
class Animal {
    void speak() {
        cout<<" speaking"<<  }

class Dog {
    void speak() {
        cout << "Barking" <<

int main() {
Dog obj;
obj.speak();
```

→ prints Barking cause speak is in class Dog. if nahi hota toh print speaking.

* Abstraction - (Implementation hiding) (e.g. email sending)

when we send mail we click send and receive pop up msg. but how data is transmitted over the network to the recipient is hidden from you.

Adv - 1) Avoids duplication
2) Reusability
3) Security.

- IO.

1) class is basically a template for objects.

this keyword in Java refers to current instance of the class.

1) Pass the current obj as parameter to another method.

2) refer to the current class instance variable.

- Parameterized constructor - (which has parameters)

```
class Student {
    string name;
    int age;

Student ( String name, int age ) {
    this.name = name;
    this.age = age; } }
```

* Exception Handling - (Technique to deal with and mechanism to recover)
An exception is an abnormal condition or error that occurs during the execution of program.

E.H is essential for proper resource management, such as closing files or releasing memory, even when an error occurs.

E.H is critical aspect of writing robust and reliable code, as it allows you to gracefully recover from errors.

(Try, catch, Throw)

↓

code that generates exception placed in 'Try' block.

* Good Hashing functions -

crucial for
this are effiecient for the operation of hash based functions data structures like hash tables or hash maps

A Good hashing fnc should have foll. characteristics.

1) Deterministic 2) Effiecient 3) Minimal Collision

Collision occurs when two diff inputs produce the same hash value.

4) Avalanche effect : 5) Deterministic output size.

Commonly used hashing functions include -

a) Dirision method - Hashing by division includes taking the remainder when dividing the input by a prime number.
e.g. (key % table_size)

b) Universal hashing - U.H uses a family of hash functions and randomly selects one at runtime.

c) Cryptographic Hash functions - SHA-256 are designed with absolute security and Avalanche effect. They are used in security-critical applications.

Hash functions are used for. Data retreival, Data integrity, Password Storage, Cryptography, Cache system, load balancing, File and Data identification.

Digital signature (Password lawun) (aadhar card)

to distribute network traffic evenly across multiple servers.

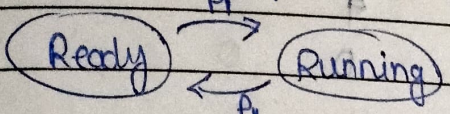
used to determine where data should be stored and retreived.

- Hashing is a technique used in c.s to store and retrieve data efficiently. It involves mapping a large data to smaller set of values using hash function. Collision can occur when two or more keys are mapped to the same hash value. linear probing is used to handle collisions. (find an empty slot and inserts the collided key there, continues untill all collided keys have been stored)

Scheduling Algorithm

Page :
Date :

Preemptive



a) SRTF b) LRTF
c) Priority d) Round Robin

Non-preemptive (whole process will execute)

Ready → Running

a) FCFS
b) SJF
c) LJF
d) Queue, multilevel queue

CPU scheduling -

Arrival Time - जेव्हा process येते Ready Queue madhe.

Burst Time - time required by process to execute on CPU.

Completion Time - Time at which process complete execution.

Turn around Time - C.T - A.T

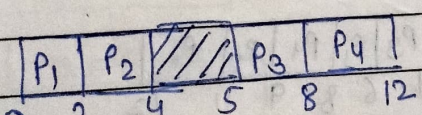
W.T = T.A.T - B.T

Response Time - (Process at which CPU get process first time) - (Arrival Time)

FCFS

Process No.	A.T	B.T	C.T	T.A.T	W.T	R.T
P_1	0	2	2	2	0	0
P_2	1	2	4	3	1	1
P_3	5	3	8	3	0	0
P_4	6	4	12	6	2	2

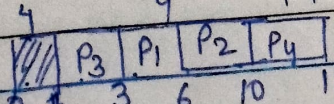
A.T dependent
Mode = N.P



SJF

Process.	A.T	B.T	C.T	T.A.T	W.T	R.T
P_1	1	3	6	5	2	2
P_2	2	4	10	8	4	4
P_3	1	2	3	2	0	0
P_4	4	4	14	10	6	6

B.T dependent
Mode = N.P.



Priority ↑ Preemptive

Priority	Process	A·T	B·T	C·T	T·A·T	W·T
10	P_1	0	5 4	12	12	7
20	P_2	1	4 3	8	7	3
30	P_3	2	2 1 0	4	2	0
40	P_4	4	1 0	5	1	0

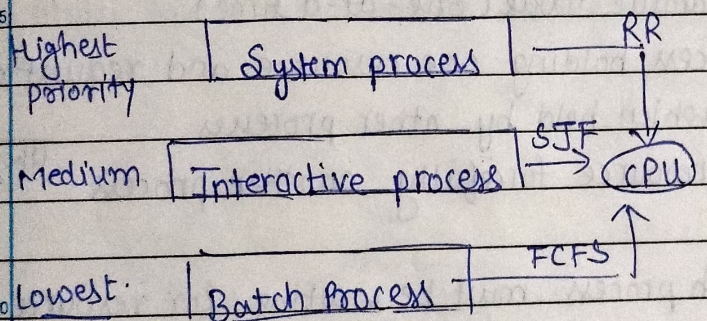
Page :
Date

P_1	P_2	P_3	P_3	P_4	P_2	P_1
0	1	2	3	4	5	8	12

at 1 P_1 and P_2 (check on priority) ($P_2 > P_1$)
at 2 P_1^4, P_2^3 and P_3^2
at 4 P_1, P_2, P_4 (check priority) ($P_4 > P_1, P_2$)

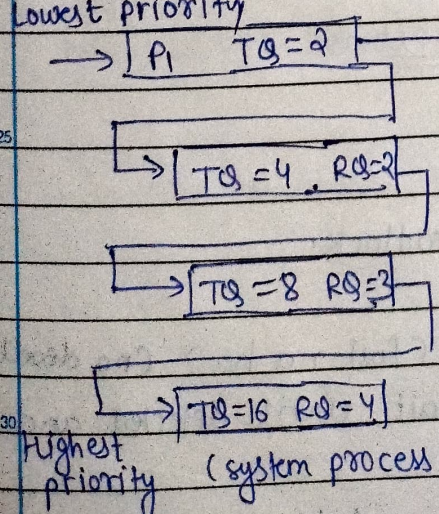
* Multilevel Queue Scheduling -

Process can be of different types., not single ready queue multiple present.



problem starvation.
If High priority process continuously coming then low priority process will wait continuously for their turn.

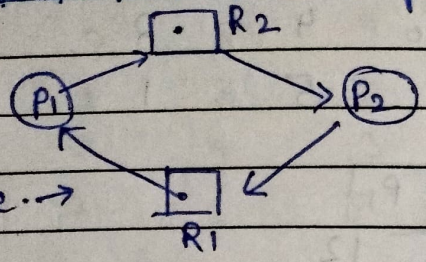
* Multilevel Feedback Queue -



$P_1 = 19\ 17\ 13\ 5\ 0$

* Deadlock - (No cycle No deadlock)
(If there is cycle & system is single instance there isn't always Deadlock)

Resource allocation Graph.



P_1 has resource R_1 and P_2 has resource R_2 but P_1 wants R_2 and P_2 wants R_1 and it is deadlock condition then infinitely waiting state.

System model -
a) Process request resource b) use resource
c) release resource.

* Necessary Condition for Deadlocks.

1) Mutual exclusion - At least one resource type in the system which can be used in non-sharable mode (one-at-a time).
2) Hold & wait - process holding one resource and requesting another resource which held by other processes.
3) No-preemption - resource forcefully another process पाएगा no take.
4) Circular wait - each process must be waiting for resource held by another process.

$P_1 \rightarrow P_2 \rightarrow P_3$

* Deadlock Handling methods -

1) Prevention - independence from deadlock.

2) Avoidance - to be in safe state (fail na ho) (no deadlock)
3) Detection & Recovery - Here we wait until detected and then recover it
4) Ignorance - ignore the problem as it doesn't exists.

We cant violate mutual exclusion to prevent Deadlock.

Page :
Date :

Deadlock Avoidance (Banker's Algorithm) also used for Deadlock detection:

Process	Allocation A B C	Max Need A B C	Available A B C	Remaining (Max - Allocation) A B C
P_1	0 1 0	7 5 3	3 3 2	7 4 3 ✓
P_2	2 0 0	3 2 2	5 3 2	1 2 2 ✓
P_3	3 0 2	9 0 2	7 4 3	6 0 0
P_4	2 1 1	4 2 2	7 4 5	2 1 1 ✓
P_5	0 0 2	5 3 3	8	5 3 1 ✓
	7 2 5		7 5 5 + 3 0 2 = 10 5 7	

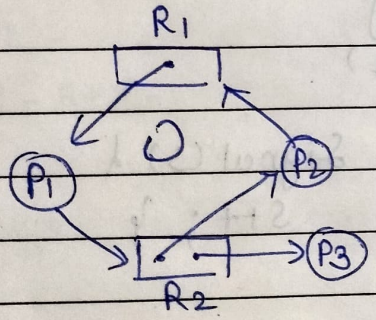
Total A = 10, B = 5, C = 7

$P_2 \rightarrow P_4 \rightarrow P_5 \rightarrow P_1 \rightarrow P_3$

Safe seq executed no deadlock

first take P_2 cause available >= remaining need then P_4 and keep adding allocation to available stuff.

Multi instance Resource Allocation -



	Allocate R1 R2	Request R1 R2
P_1	1 0	0 1
P_2	0 1	1 0
P_3	0 1	0 0

cycle but multi instance so not always Deadlock.

curr availibility (R1, R2) (0,0) + (0,1)

$P_3 \rightarrow$ terminated. = (0,1)

$P_1 \rightarrow$ request fulfilled + (1,0) = (1,1)

$P_2 \rightarrow$ fulfilled.

Mutex Lock -

```
1) while (lock = 1);
2) lock = 1
3) critical section.
4) lock = 0;        → exit code
```

initially lock = 0

first P_1 goes into critical section then P_2.

No guarantee that there will be always mutual exclusion.

5 in this case

P_1	P_2
1 7 2 3	1 2 3
(1)	CS

pre-empt (washroom gya)

10

$L = \emptyset$ 1 now P_1 comes back and override lock value and make it to 1 again $L = \emptyset$ 1 1 and goes into CS. Both are into CS now.

* Semaphore is an Integer variable used in mutual exclusive manner to achieve synchronization.

15

Semaphor

- Counting ($-\infty$ to $+\infty$)
- Binary (0, 1)

```
Wait(S) {
    while S <= 0;
    S--; }
```

```
Signal(S) {
    S++; }
```

20

Mutual exclusion implementation using semaphores.

```
do {
    waiting (mutex);    // CS
    signal (mutex);     // remainder section
} while(True);
```

25

Entry code for CS for process.

Down (Semaphore S)

S=10, then 10 processes can go into critical section

if goes (<0) then suspended.

{ S_{val} = S.value − 1;
if (S.value < 0)
{ put process (PCB) in suspended list, sleep(); }
else {
return; }

Exit code - up (semaphore S)
{
S.value = S.value+1
if (S ≤ 0)
{ select process from suspended list, wake up(); }
}

if S=10 and we perform 5P (down) and 6V (up) operations.

S=10 = 10 − 5 + 6 = 11

S = 17 - 5P, 3V, 1P

17 − 5+3−1 = 14

Deadlocks and starvation.

P_0	P_1
wait(S);	wait(Q);
wait(Q);	wait(S);
!	!
!	!
signal(S);	signal(Q);
signal(Q);	signal(S);

* Dining Philosopher Problem -

Page :
Date :

```
do {  wait ( chopstick [i] );
      wait ( chopstick [(i+1) % 5] );

   eat
      signal (chopstick [i]);
      signal (chopstick [(i+1) % 5]);
   } while (True);
```

Represent fork by Semaphore
philosopher tries to grab fork by executing wait() operation.
He releases forks by executing signal() operation.

still deadlock can occur in case that
all become hungry and grab left chopstick. and we
he will try to grab right chopstick It will be delayed
forever.

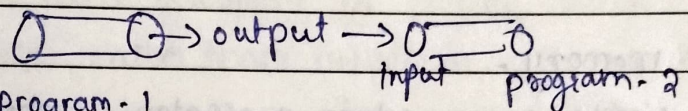
* Possible remedies
a) Allow at most 4 to sit simultaneously
b) Allow only to pick if both forks are available.
c) odd philosopher picks up first his left chop and the right and vice-versa for even philosopher.

The main purpose of OS is to execute user programs and make it easier for users to understand and interact with computers as well as run applications.

e.g. Types -

1) Batched OS (payroll sys)
2) Multi-programmed OS (Windows OS)
3) Timesharing (Multics)
4) Distributed OS (LOCUS)
5) Real-Time OS (PSOS etc).

- Pipe is a connection among two or more processes that are interrelated to each other. (interprocess communication IPC)



- Demand Paging - method that loads pages to memory on demand. used in virtual memory.

- RTOS - used for real-time applications. for those applications where data processing should be done in a fixed and small measure of time. occupies less memory and consumes fewer resources.
Types - 1) Hard RT, 2) Soft RT 3) Firm RT
e.g. Air traffic control systems., Anti-Lock Brake systems

- Process synchronization deals with coordinating the execution of multiple processes or threads to avoid conflicts. Goal is to prevent race conditions, deadlocks etc.

- IPC (interprocess communication) is simply used for exchanging data between multiple threads in one or more programs diff. processes can communicate with each other with approval of OS. e.g. Pipes, semaphores, signals

Primary Memory	Secondary memory.
a) RAM	a) Hard disks, USB, CDs etc.
b) costly	b) less costly
c) data stored is temporarily	c) permanently
d) data lost when power failure.	
e) accessed by data.	e) accessed by I/O channels.

- overlays in OS is basically a programming method that divides processes into pieces so that the instructions are imp. and need to be saved in memory. does not need any type of support from OS.

Q. Virtual memory.

→ It is simply space where a greater number of programs can be stored by themselves in form of pages. can be manged by two common ways by OS ie. Paging and Segmentation.

It acts as a temporary storage that can be used along with RAM for computer processes.

Q. Threads.

→ Basic unit of CPU utilization that makes communication more effective and effiecient. are called lightweight processes cause they have their own stack but can access shared data.

Each thread has its own Program counter, Registers, Stack and State.

- Process is a program that is currently under execution. when a program is loaded into memory it becomes process. Can be divided into four sections - stack, heap, text and data.

Two processes -

a) OS processes b) User processes

v-v-vice versa

Page :
Date :

Paging	Segmentation
a) non-contiguous allocation technique that divides process in form of pages.	a) divides processes into modules and parts of different sizes
b) invisible to programmer	b) v-v
c) sized fixed	c) v-v
d) available on CPUs	d) windows server.
e) internal fragmentation	e) External

- Thrashing is a situation where CPU performs less productive work and more swapping or paging work. spends more time on this rather than execution. It occurs when the process does not have enough pages due to which page fault rate is increased.

Multiprogramming is to have at least some processes running at all times. improves utilization of the CPU as it organizes many jobs where the CPU always has one to execute.

Q. Asymmetric clustering.

⇒ system in which one of the nodes among all nodes is in hot standby mode whereas the rest of all nodes run diff. programs. (applications) uses all hardware resources therefore considered more reliable system as compared to others.

Multiprocessing - system that allows to process two or more diff. portions of the same program simultaneously. less economical, efficient, no. of CPU's more than one.

- Sockets in OS are generally referred to as endpoint for IPC. Here endpoint is referred to as IP address + Port number. Used in client-server based systems. Sockets are used to make it easy for software developers to create network enabled programs.

Types - a) Stream b) Datagram c) Raw

Q. Kernel.

→ central component OS. system starts it is loaded first in main memory. interface btw applications and hardware. Kernel is how a system starts up, translates input and outputs requests to the CPU. Each os have seperate kernel.

Q. Context Switching.

→ process of saving the context of one process and loading the context of another process. coest effective and time-saving. cause it allows multiple processes to share a single CPU.

- There are basically 4 sections of process.

Stack - used for local variables and returns addresses.
↕
Heap - used for dynamic memory allocation.
Data - Stores global and static variables.
Text - comprises compiled program code.

- Belady anomaly is a phenomenon in which if we increase the number of frames in memory, the no. of page fault also ↑. It is generally experienced when we use FIFO page replacement Algorithm.

- Spooling stands for simultaneous peripheral operations online. referred as putting data of various I/O jobs in a buffer. It is used for mediation between a computer application and a slow peripheral. very useful cause devices access or acquire data at different rates.

Multilevel scheduling = round robin + priority scheduling.

Page :
Date :

thrashing - ↑ degree of multiprogramming → ↑ CPU works then thrashing occurs, rather than executing programs it only loads page-replacement Algorithm.

Booting Process - (Process done after pressing the button and before loading the OS)

→ POWER ON → Basic I/O system.
→ CPU will move to BIOS in ROM.
→ BIOS will be executed. (POST)
→ Power on Self Test. (All the hardware will be tested).
→ BIOS will load MBR to RAM
↳ master boot record.
→ MBR will load bootloader to RAM.
→ bootloader will load OS to RAM.

Soft Booting - Restart
Hard Booting - Power ON

Process	Threads
a) Independent	a) subset of processes
b) seperate memory addr.	b) share their add space.
c) context switch slow	c) faster.

* Multithreading -
Multiple threads exist in one process and each share data and memory space btw all threads in the process.

NodeJS (server side language)

supports CommonJs and ES6 modules for code organization and reusability.

Page :
Date :

- ExpressJS popular NodeJS framework for building web applications provides functionality for routing, middleware and error handling. Restful API can be built using Express.Js

- Seeders can be used to manage database changes.
- JSON web tokens and Passport.Js can be used for authorization and authentication.

ExpressJs

↳ * Routing.-

Routing is the process of determining what should happen on calling a URL. typically done using ExpressJs. Routes are created using HTTP methods like POST, PUT, GET etc.

used to create robust & efficient app in NodeJs

* Middleware.-

Can be used to handle specific tasks for a route. Middleware func have access to request and response objects in NodeJS. They can use to perform tasks like logging, authentication and error handling.

* Streaming -

Streams are used for efficient data processing. They allow you to read data from a source or write data to destination in a continuous manner.

used for data processing NodeJs

Buffers are used to handle binary data and transform streams. They are useful for reading and processing large datasets.

* Mocha and Chai can be used to test your NodeJS applications. Apache Bench and Artillery can used for test performance of app

Passport.JS is an authentication middleware for NodeJS that is designed to be extremely adaptable and modular, allowing users to integrate different authentication strategies into their applications

used for authentication and authorization for NodeJS application

Json web tokens is a compact and self-contained way of securely transmitting information btw parties as JSON object. contains three parts. a) header
b) Payload
c) Signature

header.payload.signature

Asynchronous programming in Node can be performed using callbacks, promises and async/await. allows user to write non-blocking code that can handle multiple requests. It includes callbacks (fnc that are passed as argument to other funct and are executed after the completion of the parent function. Promises are objects that represent the eventual completion of an asynchronous operation and allow you to chain multiple asyn operations together.

- Core modules in NodeJS -

1) File system - functionality such as reading and writing files, creating directories and deleting files.
2) http and https - allow to create web servers and HTTP requests, handling requests and responses and parsing URLs.
3) events - event emitter that allow you to create custom events. event listener is main functionality.
4) util - utility functions for common tasks such as formatting strings.

Many NoSQL databases have large datasets capacity to store data in JSON format, which is native to Javascript. → fast read & write opr.

LINUX

Page :
Date :

- Core of LINUX - kernel.
- Binary Directory - /bin, /sbin, /lib, /opt.
- Directory is a type of file.
- / represents top level directory.
- ~ rep. user home directory.
- wc used to count total no. of lines, words & chrc.
- rm used to remove files.
- x used to delete single character.
- pwd used to know which directory u are in
- ls - a used to see all hidden files
- ls - F give full listing.
- touch use to create new file.
- cp used to copy files through command line.
- mv abc.txt pqr.txt to rename file.
- locate command use for search.
- locate - i * hello * this to locate file with words hello & this
- cat used to display content of file.
- sudo stands for superuser do.
- df use to see the available disk space.
- hostname - I gives you your IP address
- oss stands for open source Software.
- sync use to force all buffers to disk.
- xcat use to view zipped files
- bunzip2 decompresses a file
- script used to record a user login session.
- uname use to display the OS name.
- lp use to print a file.
- lp - d [printer_name] [path or filename] print the file using the default printer in LINUX.
- lp - d [printer_name] - n 5 [path or filename] with multiple copies.
- 3 types of users in LINUX (super users, system users & regular/ normal users)

Redirection opr

- > erasing all existing data of that file.
- >> without erasing.

CTRL + C / CTRL + Pause/Break terminate execution

Camlin

GIT

clone - git clone <- some link ->

status - git status

- when we modify any file then we have two steps add → commit
- In git status.
  untracked - new files that git doesn't yet track.
  modified - changed.
  staged - file is ready to be committed.
  unmodified - unchanged.

change (modified) / new file (untracked)
↓
add (staged)
↓
commit.

- git add. → to git staging area. means add all files. (ready to commit)
- git commit -m "some message" (ready to be push on our github repo)
- git push origin main.
  ↳ (upload local repo to remote repo)

- mkdir (make new directory)
  to intialized git we use git init.

- to push our files into some new repo. we use
  git remote add origin <- link ->
  git branch -m main (to rename branch) then we
- git push origin main. or git push -u origin main
  ↓
  if we are working on same project for long time then all the commits will be in origin main.

* workflow
  repo → clone → changes → add → commit → push

* Merge - (two branches)
  git diff <- branch name -> then git merge <- branch name ->

MONGODB. (No SQL DB)

can insert entry through our shell like

- db.books.insertone({...})
  ↳ collection

also can insert many documents like

- db.books.insertMany([{...},{...}])

- if we write db.books.find() by default it will give first 20 books then if we write it it will give another 20.

- db.books.find({author: "Rugs"}, {title:1, author:1})
  it will give books with only title and author.

- db.books.find().count() (gives the total count)
- db.books.find().sort({title:-1})
  ↳ will sort in descending order.

we can nest loops in our doc. mhanje infor chya madhe information takli.

- db.books.find({rating:{$gt:7}})
  (find books of rating >7)

- can use or operator by $or.
  db.finder books. ({$or, [{rating:7}, {rating:9}]})

db.books.find({rating: {$in: [7,8,9]}})
(find boks with rating 7,8 or 9.

$nin gives opposite.

- Quering arrays.
  db.books.find({genres: ["fantasy"]})
  (return book with genre only as fantasy)

  .find({genres: {$all: ["fantasy", "sci-fi"]}})
  (have fantasy as well as sci-fi as genre)

- db.books.find({"reviews.name": "luigi"})
  (reviews madhe name = luigi asel tr karan review madhe nested array aahe (name & body)

- delete works same as insert

- db.books.updateOne({_id: unique-id}, {$set: {rating: 8.8, pages: 376}})

- $inc: {pages: 2} (will increment pages by 2 for that obj)

- $push - pushing new item into obj array.
  $pull - removing item from obj array.

| $each | allows us to push several items to genres (which is array already)

MongoDB drivers.

In app.js.

```
const expres = require('express') // import package from node modules
const app = express()

app.listen(3000, () => {
  console.log('app listening on port 3000')
})
// routes
app.get('/books', (req, res) => {
  res.json({msg: "welcome pappu"})
})
```

- Connecting to MongoDb

  we have a db file. which exports 2 function 1 is connecttodb and 2nd is get (return db connection) and we invoke this fnc in app.js files and now we have db obj we can use in our diff endpoint handling fnctions.

- in shell we used db.books.find()
  in node we use db.collection('books').find()

```
let book[]
db.collection('books').find().sort({author:1})
.forEach(book => books.push(book)).then(()=>
{ res.status(200).json(books) })
.catch(() => { res.status(500).json({
    error: 'could not fetch data'
```

Handling post req.

```
app.post('/books', (req, res) => {
  const book = req.body

db.collection('books').insertOne(book)
  .then(result => {
      res.status(201).json(result) })
  .catch(err => {
      res.status(500).json({err: '...'})})
```

- delete req works same as get request.

```
app.delete('/books/:id', (req, res) => {
  if(ObjectId.isValid(req.params.id)){
    db.collection('books')
    .deleteOne({_id: ObjectId(req.params.id)})
  .then( result => {
    res.status(200).json(result) })
  .catch(err => {
    res.status(500).json({error: '...'})
    })
  else{
      (500).json({error: '...'})})
```

- patch works as post req.

```
  const updates = req.body
```

pagination.

const page = req.query.p || 0 → default page

const booksPerPage = 3

- skip(page * booksPerPage)
- limit(booksPerPage)

if page = 1 will skip first 3 pages and show to another

* Indexes -

db.books.find({rating: 8}) is inefficient cause we check for every document among a large dataset (suppose 2|5)
instead what we do is createInden.

db.books.createIndex({rating: 8})
create(rating_8)

and then we do
db.books.getIndexes() (we get only 2 so we go through document of having rating only 8)

- MongoDB, NoSQL, high scalability, flexibility
mongoDB stores data in JSON document structure form
- Auto sharding in MongoDB distributes data across multiple servers, enabling horizontal scaling and efficient handling of large datasets.
- high availibility, providing fault tolerance, maintaining multiple copies ensure data redundancy.