



**Φ Zerone** - Stands for **Zero (0)** and **One (1)**, the very basics of computing. So Binary Matters. Infact only Binary Matters in our world of computers. The motive behind Zerone is to make job hunting an easier process or atleast that is what we intent to. This is a one stop jobs group. The content in this document is collected from various groups, sites & media. Copyright belongs to respective owners. If you have any copyright issues, please mail to [legal@zeroneworld.com](mailto:legal@zeroneworld.com) with complete details.

**"If You Have An Apple And I Have An Apple And We Exchange Apples Then You And I Will Still Each Have One Apple. But If You Have An Idea And I Have An Idea And We Exchange These Ideas, Then Each Of Us Will Have Two Ideas."**  
--- **George Bernard Shaw (1856-1950)**

Visit group at <http://groups.google.com/group/zerone01>  
Visit: <http://www.ZeroneWorld.com>

# UNIX Concepts

## SECTION - I

### FILE MANAGEMENT IN UNIX

#### 1. *How are devices represented in UNIX?*

All devices are represented by files called *special files* that are located in `/dev` directory. Thus, device files and other files are named and accessed in the same way. A 'regular file' is just an ordinary data file in the disk. A 'block special file' represents a device with characteristics similar to a disk (data transfer in terms of blocks). A 'character special file' represents a device with characteristics similar to a keyboard (data transfer is by stream of bits in sequential order).

#### 1. *What is 'inode'?*

All UNIX files have its description stored in a structure called 'inode'. The inode contains info about the file-size, its location, time of last access, time of last modification, permission and so on. Directories are also represented as files and have an associated inode. In addition to descriptions about the file, the inode contains pointers to the data blocks of the file. If the file is large, inode has indirect pointer to a block of pointers to additional data blocks (this further aggregates for larger files). A block is typically 8k.

Inode consists of the following fields:

- Ø File owner identifier
- Ø File type
- Ø File access permissions
- Ø File access times
- Ø Number of links
- Ø File size
- Ø Location of the file data

#### 1. *Brief about the directory representation in UNIX*

A Unix directory is a file containing a correspondence between filenames and inodes. A directory is a special file that the kernel maintains. Only kernel modifies directories, but processes can read directories. The contents of a directory are a list of filename and inode number pairs. When new directories are created, kernel makes two entries named '.' (refers to the directory itself) and '..' (refers to parent directory). System call for creating directory is `mkdir (pathname, mode)`.

#### 1. *What are the Unix system calls for I/O?*

- Ø `open(pathname,flag,mode)` - open file
- Ø `creat(pathname,mode)` - create file
- Ø `close(filedes)` - close an open file
- Ø `read(filedes,buffer,bytes)` - read data from an open file
- Ø `write(filedes,buffer,bytes)` - write data to an open file
- Ø `lseek(filedes,offset,from)` - position an open file
- Ø `dup(filedes)` - duplicate an existing file descriptor
- Ø `dup2(oldfd,newfd)` - duplicate to a desired file descriptor
- Ø `fcntl(filedes,cmd,arg)` - change properties of an open file

Ø `ioctl(filedes,request,arg)` - change the behaviour of an open file  
The difference between `fcntl` and `ioctl` is that the former is intended for any open file, while the latter is for device-specific operations.

### 1. How do you change File Access Permissions?

Every file has following attributes:

- Ø owner's user ID ( 16 bit integer )
- Ø owner's group ID ( 16 bit integer )
- Ø File access mode word  
'r w x -r w x- r w x'  
(user permission-group permission-others permission)  
r-read, w-write, x-execute

To change the access mode, we use `chmod(filename,mode)`.

Example 1:

To change mode of myfile to 'rw-rw-r--' (ie. read, write permission for user - read,write permission for group - only read permission for others) we give the args as:

```
chmod(myfile,0664) .
```

Each operation is represented by discrete values

- 'r' is 4
- 'w' is 2
- 'x' is 1

Therefore, for 'rw' the value is 6(4+2).

Example 2:

To change mode of myfile to 'rwxr--r--' we give the args as:  
`chmod(myfile,0744)`.

### 1. What are links and symbolic links in UNIX file system?

A link is a second name (not a file) for a file. Links can be used to assign more than one name to a file, but cannot be used to assign a directory more than one name or link filenames on different computers.

Symbolic link 'is' a file that only contains the name of another file. Operation on the symbolic link is directed to the file pointed by the it. Both the limitations of links are eliminated in symbolic links.

Commands for linking files are:

- Link `ln filename1 filename2`
- Symbolic link `ln -s filename1 filename2`

### 1. What is a FIFO?

FIFO are otherwise called as 'named pipes'. FIFO (first-in-first-out) is a special file which is said to be data transient. Once data is read from named pipe, it cannot be read again. Also, data can be read only in the order written. It is used in interprocess communication where a process writes to one end of the pipe (producer) and the other reads from the other end (consumer).

### 1. How do you create special files like named pipes and device files?

The system call `mknod` creates special files in the following sequence.

1. kernel assigns new inode,
1. sets the file type to indicate that the file is a pipe, directory or special file,
1. If it is a device file, it makes the other entries like major, minor device numbers.

For example:

If the device is a disk, major device number refers to the disk controller and minor device number is the disk.

1. *Discuss the mount and unmount system calls*

The privileged mount system call is used to attach a file system to a directory of another file system; the unmount system call detaches a file system. When you mount another file system on to your directory, you are essentially splicing one directory tree onto a branch in another directory tree. The first argument to mount call is the mount point, that is, a directory in the current file naming system. The second argument is the file system to mount to that point. When you insert a cdrom to your unix system's drive, the file system in the cdrom automatically mounts to /dev/cdrom in your system.

1. *How does the inode map to data block of a file?*

Inode has 13 block addresses. The first 10 are direct block addresses of the first 10 data blocks in the file. The 11th address points to a one-level index block. The 12th address points to a two-level (double in-direction) index block. The 13th address points to a three-level (triple in-direction) index block. This provides a very large maximum file size with efficient access to large files, but also small files are accessed directly in one disk read.

1. *What is a shell?*

A shell is an interactive user interface to an operating system services that allows an user to enter commands as character strings or through a graphical user interface. The shell converts them to system calls to the OS or forks off a process to execute the command. System call results and other information from the OS are presented to the user through an interactive interface. Commonly used shells are sh, csh, ks etc.

## SECTION - II

### PROCESS MODEL and IPC

1. *Brief about the initial process sequence while the system boots up.*

While booting, special process called the 'swapper' or 'scheduler' is created with Process-ID 0. The swapper manages memory allocation for processes and influences CPU allocation. The swapper inturn creates 3 children:

Ø the process dispatcher,

Ø vhand and

Ø dbflush

with IDs 1,2 and 3 respectively.

This is done by executing the file /etc/init. Process dispatcher gives birth to the shell. Unix keeps track of all the processes in an internal data structure called the Process Table (listing command is ps -el).

2. *What are various IDs associated with a process?*

Unix identifies each process with a unique integer called ProcessID. The process that executes the request for creation of a process is called the 'parent process' whose PID is 'Parent Process ID'. Every process is associated with a particular user called the 'owner' who has privileges over the process. The identification for the user

is 'UserID'. Owner is the user who executes the process. Process also has 'Effective User ID' which determines the access privileges for accessing resources like files.

getpid() -process id  
getppid() -parent process id  
getuid() -user id  
geteuid() -effective user id

3. *Explain fork() system call.*

The `fork()` used to create a new process from an existing process. The new process is called the child process, and the existing process is called the parent. We can tell which is which by checking the return value from `fork()`. The parent gets the child's pid returned to him, but the child gets 0 returned to him.

4. *Predict the output of the following program code*

```
main()
{
    fork();
    printf("Hello World!");
}
```

**Answer:**

Hello World!Hello World!

**Explanation:**

The fork creates a child that is a duplicate of the parent process. The child begins from the fork().All the statements after the call to fork() will be executed twice.(once by the parent process and other by child). The statement before fork() is executed only by the parent process.

5. *Predict the output of the following program code*

```
main()
{
    fork(); fork(); fork();
    printf("Hello World!");
}
```

**Answer:**

"Hello World" will be printed 8 times.

**Explanation:**

$2^n$  times where n is the number of calls to fork()

6. *List the system calls used for process management:*

<b>System calls</b>	<b>Description</b>
fork()	To create a new process
exec()	To execute a new program in a process
wait()	To wait until a created process completes its execution
exit()	To exit from a process execution
getpid()	To get a process identifier of the current process
getppid()	To get parent process identifier
nice()	To bias the existing priority of a process
brk()	To increase/decrease the data segment size of a process

7. *How can you get/set an environment variable from a program?*

Getting the value of an environment variable is done by using ``getenv()'`.  
Setting the value of an environment variable is done by using ``putenv()'`.

8. *How can a parent and child process communicate?*

A parent and child can communicate through any of the normal inter-process communication schemes (pipes, sockets, message queues, shared memory), but also have some special ways to communicate that take advantage of their relationship as a parent and child. One of the most obvious is that the parent can get the exit status of the child.

9. *What is a zombie?*

When a program forks and the child finishes before the parent, the kernel still keeps some of its information about the child in case the parent might need it - for example, the parent may need to check the child's exit status. To be able to get this information, the parent calls ``wait()'`; In the interval between the child terminating and the parent calling ``wait()'`, the child is said to be a ``zombie'` (If you do ``ps'`, the child will have a ``Z'` in its status field to indicate this.)

10. *What are the process states in Unix?*

As a process executes it changes state according to its circumstances. Unix processes have the following states:

Running : The process is either running or it is ready to run .

Waiting : The process is waiting for an event or for a resource.

Stopped : The process has been stopped, usually by receiving a signal.

Zombie : The process is dead but have not been removed from the process table.

11. *What Happens when you execute a program?*

When you execute a program on your UNIX system, the system creates a special environment for that program. This environment contains everything needed for the system to run the program as if no other program were running on the system. Each process has process context, which is everything that is unique about the state of the program you are currently running. Every time you execute a program the UNIX system does a fork, which performs a series of operations to create a process context and then execute your program in that context. The steps include the following:

Ø Allocate a slot in the process table, a list of currently running programs kept by UNIX.

Ø Assign a unique process identifier (PID) to the process.

Ø iCopy the context of the parent, the process that requested the spawning of the new process.

Ø Return the new PID to the parent process. This enables the parent process to examine or control the process directly.

After the fork is complete, UNIX runs your program.

12. *What Happens when you execute a command?*

When you enter ``ls'` command to look at the contents of your current working directory, UNIX does a series of things to create an environment for ``ls'` and the run it: The shell has UNIX perform a fork. This creates a new process that the shell will use to run the ``ls'` program. The shell has UNIX perform an ``exec'` of the ``ls'` program. This

replaces the shell program and data with the program and data for ls and then starts running that new program. The ls program is loaded into the new process context, replacing the text and data of the shell. The ls program performs its task, listing the contents of the current directory.

### 13. What is a Daemon?

A daemon is a process that detaches itself from the terminal and runs, disconnected, in the background, waiting for requests and responding to them. It can also be defined as the background process that does not belong to a terminal session. Many system functions are commonly performed by daemons, including the sendmail daemon, which handles mail, and the NNTP daemon, which handles USENET news. Many other daemons may exist. Some of the most common daemons are:

- Ø init: Takes over the basic running of the system when the kernel has finished the boot process.
- Ø inetd: Responsible for starting network services that do not have their own stand-alone daemons. For example, inetd usually takes care of incoming rlogin, telnet, and ftp connections.
- Ø cron: Responsible for running repetitive tasks on a regular schedule.

### 14. What is 'ps' command for?

The ps command prints the process status for some or all of the running processes. The information given are the process identification number (PID), the amount of time that the process has taken to execute so far etc.

### 15. How would you kill a process?

The kill command takes the PID as one argument; this identifies which process to terminate. The PID of a process can be got using 'ps' command.

### 16. What is an advantage of executing a process in background?

The most common reason to put a process in the background is to allow you to do something else interactively without waiting for the process to complete. At the end of the command you add the special background symbol, &. This symbol tells your shell to execute the given command in the background.

Example: cp \*.\* ../backup& (cp is for copy)

### 17. How do you execute one program from within another?

The system calls used for low-level process creation are execlp() and execvp(). The execlp call overlays the existing program with the new one, runs that and exits. The original program gets back control only when an error occurs.

execlp(path,file\_name,arguments..); //last argument must be NULL

A variant of execlp called execvp is used when the number of arguments is not known in advance.

execvp(path,argument\_array); //argument array should be terminated by NULL

### 18. What is IPC? What are the various schemes available?

The term IPC (Inter-Process Communication) describes various ways by which different process running on some operating system communicate between each other. Various schemes available are as follows:

*Pipes:*

One-way communication scheme through which different process can communicate. The problem is that the two processes should have a common ancestor (parent-child relationship). However this problem was fixed with the introduction of named-pipes (FIFO).

*Message Queues :*

Message queues can be used between related and unrelated processes running on a machine.

*Shared Memory:*

This is the fastest of all IPC schemes. The memory to be shared is mapped into the address space of the processes (that are sharing). The speed achieved is attributed to the fact that there is no kernel involvement. But this scheme needs synchronization.

Various forms of synchronisation are mutexes, condition-variables, read-write locks, record-locks, and semaphores.

## SECTION - III

### MEMORY MANAGEMENT

1. *What is the difference between Swapping and Paging?*

*Swapping:*

Whole process is moved from the swap device to the main memory for execution. Process size must be less than or equal to the available main memory. It is easier to implementation and overhead to the system. Swapping systems does not handle the memory more flexibly as compared to the paging systems.

*Paging:*

Only the required memory pages are moved to main memory from the swap device for execution. Process size does not matter. Gives the concept of the virtual memory.

It provides greater flexibility in mapping the virtual address space into the physical memory of the machine. Allows more number of processes to fit in the main memory simultaneously. Allows the greater process size than the available physical memory. Demand paging systems handle the memory more flexibly.

1. *What is major difference between the Historic Unix and the new BSD release of Unix System V in terms of Memory Management?*

Historic Unix uses Swapping – entire process is transferred to the main memory from the swap device, whereas the Unix System V uses Demand Paging – only the part of the process is moved to the main memory. Historic Unix uses one Swap Device and Unix System V allow multiple Swap Devices.

1. *What is the main goal of the Memory Management?*

- Ø It decides which process should reside in the main memory,
- Ø Manages the parts of the virtual address space of a process which is non-core resident,
- Ø Monitors the available main memory and periodically write the processes into the swap device to provide more processes fit in the main memory simultaneously.

1. *What is a Map?*

A Map is an Array, which contains the addresses of the free space in the swap device that are allocatable resources, and the number of the resource units available there.

Address	Units
1	10,000

This allows First-Fit allocation of contiguous blocks of a resource. Initially the Map contains one entry – address (block offset from the starting of the swap area) and the total number of resources.

Kernel treats each unit of Map as a group of disk blocks. On the allocation and freeing of the resources Kernel updates the Map for accurate information.

1. *What scheme does the Kernel in Unix System V follow while choosing a swap device among the multiple swap devices?*

Kernel follows Round Robin scheme choosing a swap device among the multiple swap devices in Unix System V.

1. *What is a Region?*

A Region is a continuous area of a process's address space (such as text, data and stack). The kernel in a 'Region Table' that is local to the process maintains region. Regions are sharable among the process.

1. *What are the events done by the Kernel after a process is being swapped out from the main memory?*

When Kernel swaps the process out of the primary memory, it performs the following:

- Ø Kernel decrements the Reference Count of each region of the process. If the reference count becomes zero, swaps the region out of the main memory,
- Ø Kernel allocates the space for the swapping process in the swap device,
- Ø Kernel locks the other swapping process while the current swapping operation is going on,
- Ø The Kernel saves the swap address of the region in the region table.

1. *Is the Process before and after the swap are the same? Give reason.*

Process before swapping is residing in the primary memory in its original form. The regions (text, data and stack) may not be occupied fully by the process, there may be few empty slots in any of the regions and while swapping Kernel do not bother about the empty slots while swapping the process out.

After swapping the process resides in the swap (secondary memory) device. The regions swapped out will be present but only the occupied region slots but not the empty slots that were present before assigning.

While swapping the process once again into the main memory, the Kernel referring to the Process Memory Map, it assigns the main memory accordingly taking care of the empty slots in the regions.

1. *What do you mean by u-area (user area) or u-block?*

This contains the private data that is manipulated only by the Kernel. This is local to the Process, i.e. each process is allocated a u-area.

1. *What are the entities that are swapped out of the main memory while swapping the process out of the main memory?*

All memory space occupied by the process, process's u-area, and Kernel stack are swapped out, theoretically.

Practically, if the process's u-area contains the Address Translation Tables for the process then Kernel implementations do not swap the u-area.

1. *What is Fork swap?*

fork() is a system call to create a child process. When the parent process calls fork() system call, the child process is created and if there is short of memory then the child process is sent to the read-to-run state in the swap device, and return to the user state without swapping the parent process. When the memory will be available the child process will be swapped into the main memory.

1. *What is Expansion swap?*

At the time when any process requires more memory than it is currently allocated, the Kernel performs Expansion swap. To do this Kernel reserves enough space in the swap device. Then the address translation mapping is adjusted for the new virtual address space but the physical memory is not allocated. At last Kernel swaps the process into the assigned space in the swap device. Later when the Kernel swaps the process into the main memory this assigns memory according to the new address translation mapping.

1. *How the Swapper works?*

The swapper is the only process that swaps the processes. The Swapper operates only in the Kernel mode and it does not uses System calls instead it uses internal Kernel functions for swapping. It is the archetype of all kernel process.

1. *What are the processes that are not bothered by the swapper? Give Reason.*

- Ø Zombie process: They do not take any up physical memory.
- Ø Processes locked in memories that are updating the region of the process.
- Ø Kernel swaps only the sleeping processes rather than the 'ready-to-run' processes, as they have the higher probability of being scheduled than the Sleeping processes.

1. *What are the requirements for a swapper to work?*

The swapper works on the highest scheduling priority. Firstly it will look for any sleeping process, if not found then it will look for the ready-to-run process for swapping. But the major requirement for the swapper to work the ready-to-run process must be core-resident for at least 2 seconds before swapping out. And for swapping in the process must have been resided in the swap device for at least 2 seconds. If the requirement is not satisfied then the swapper will go into the wait state on that event and it is awoken once in a second by the Kernel.

1. *What are the criteria for choosing a process for swapping into memory from the swap device?*

The resident time of the processes in the swap device, the priority of the processes and the amount of time the processes had been swapped out.

1. *What are the criteria for choosing a process for swapping out of the memory to the swap device?*

- Ø The process's memory resident time,
- Ø Priority of the process and
- Ø The nice value.

1. *What do you mean by nice value?*

Nice value is the value that controls {increments or decrements} the priority of the process. This value that is returned by the nice () system call. The equation for using nice value is:

$$\text{Priority} = (\text{"recent CPU usage"}/\text{constant}) + (\text{base-priority}) + (\text{nice value})$$

Only the administrator can supply the nice value. The nice () system call works for the running process only. Nice value of one process cannot affect the nice value of the other process.

1. *What are conditions on which deadlock can occur while swapping the processes?*

- Ø All processes in the main memory are asleep.
- Ø All 'ready-to-run' processes are swapped out.
- Ø There is no space in the swap device for the new incoming process that are swapped out of the main memory.
- Ø There is no space in the main memory for the new incoming process.

1. *What are conditions for a machine to support Demand Paging?*

- Ø Memory architecture must based on Pages,
- Ø The machine must support the 'restartable' instructions.

1. *What is 'the principle of locality'?*

It's the nature of the processes that they refer only to the small subset of the total data space of the process. i.e. the process frequently calls the same subroutines or executes the loop instructions.

1. *What is the working set of a process?*

The set of pages that are referred by the process in the last 'n', references, where 'n' is called the *window* of the working set of the process.

1. *What is the window of the working set of a process?*

The window of the working set of a process is the total number in which the process had referred the set of pages in the working set of the process.

1. *What is called a page fault?*

Page fault is referred to the situation when the process addresses a page in the working set of the process but the process fails to locate the page in the working set. And on a page fault the kernel updates the working set by reading the page from the secondary device.

1. *What are data structures that are used for Demand Paging?*

Kernel contains 4 data structures for Demand paging. They are,

- Ø Page table entries,
- Ø Disk block descriptors,
- Ø Page frame data table (pfdata),
- Ø Swap-use table.

26. What are the bits that support the demand paging?

Valid, Reference, Modify, Copy on write, Age. These bits are the part of the page table entry, which includes physical address of the page and protection bits.

Page address	Age	Copy on write	Modify	Reference	Valid	Protection
--------------	-----	---------------	--------	-----------	-------	------------

26. How the Kernel handles the fork() system call in traditional Unix and in the System V Unix, while swapping?

Kernel in traditional Unix, makes the duplicate copy of the parent's address space and attaches it to the child's process, while swapping. Kernel in System V Unix, manipulates the region tables, page table, and pfdata table entries, by incrementing the reference count of the region table of shared regions.

26. Difference between the fork() and vfork() system call?

During the fork() system call the Kernel makes a copy of the parent process's address space and attaches it to the child process.

But the vfork() system call do not makes any copy of the parent's address space, so it is faster than the fork() system call. The child process as a result of the vfork() system call executes exec() system call. The child process from vfork() system call executes in the parent's address space (this can overwrite the parent's data and stack ) which suspends the parent process until the child process exits.

26. What is BSS(Block Started by Symbol)?

A data representation at the machine level, that has initial values when a program starts and tells about how much space the kernel allocates for the uninitialized data. Kernel initializes it to zero at run-time.

26. What is Page-Stealer process?

This is the Kernel process that makes rooms for the incoming pages, by swapping the memory pages that are not the part of the working set of a process. Page-Stealer is created by the Kernel at the system initialization and invokes it throughout the lifetime of the system. Kernel locks a region when a process faults on a page in the region, so that page stealer cannot steal the page, which is being faulted in.

26. Name two paging states for a page in memory?

The two paging states are:

- Ø The page is aging and is not yet eligible for swapping,
- Ø The page is eligible for swapping but not yet eligible for reassignment to other virtual address space.

26. What are the phases of swapping a page from the memory?

- Ø Page stealer finds the page eligible for swapping and places the page number in the list of pages to be swapped.

- Ø Kernel copies the page to a swap device when necessary and clears the *valid* bit in the page table entry, decrements the pfddata reference count, and places the pfddata table entry at the end of the free list if its reference count is 0.

33. *What is page fault? Its types?*

Page fault refers to the situation of not having a page in the main memory when any process references it.

There are two types of page fault :

- Ø Validity fault,
- Ø Protection fault.

33. *In what way the Fault Handlers and the Interrupt handlers are different?*

Fault handlers are also an interrupt handler with an exception that the interrupt handlers cannot sleep. Fault handlers sleep in the context of the process that caused the memory fault. The fault refers to the running process and no arbitrary processes are put to sleep.

33. *What is validity fault?*

If a process referring a page in the main memory whose valid bit is not set, it results in validity fault.

The valid bit is not set for those pages:

- Ø that are outside the virtual address space of a process,
- Ø that are the part of the virtual address space of the process but no physical address is assigned to it.

33. *What does the swapping system do if it identifies the illegal page for swapping?*

If the disk block descriptor does not contain any record of the faulted page, then this causes the attempted memory reference is invalid and the kernel sends a “*Segmentation violation*” signal to the offending process. This happens when the swapping system identifies any invalid memory reference.

33. *What are states that the page can be in, after causing a page fault?*

- Ø On a swap device and not in memory,
- Ø On the free page list in the main memory,
- Ø In an executable file,
- Ø Marked “demand zero”,
- Ø Marked “demand fill”.

33. *In what way the validity fault handler concludes?*

- Ø It sets the valid bit of the page by clearing the modify bit.
- Ø It recalculates the process priority.

33. *At what mode the fault handler executes?*

At the Kernel Mode.

33. *What do you mean by the protection fault?*

Protection fault refers to the process accessing the pages, which do not have the access permission. A process also incur the protection fault when it attempts to write a page whose *copy on write* bit was set during the fork() system call.

33. *How the Kernel handles the copy on write bit of a page, when the bit is set?*

In situations like, where the copy on write bit of a page is set and that page is shared by more than one process, the Kernel allocates new page and copies the content to the new page and the other processes retain their references to the old page. After copying the Kernel updates the page table entry with the new page number. Then Kernel decrements the reference count of the old pfddata table entry.

In cases like, where the copy on write bit is set and no processes are sharing the page, the Kernel allows the physical page to be reused by the processes. By doing so, it clears the copy on write bit and disassociates the page from its disk copy (if one exists), because other process may share the disk copy. Then it removes the pfddata table entry from the page-queue as the new copy of the virtual page is not on the swap device. It decrements the swap-use count for the page and if count drops to 0, frees the swap space.

33. *For which kind of fault the page is checked first?*

The page is first checked for the validity fault, as soon as it is found that the page is invalid (valid bit is clear), the validity fault handler returns immediately, and the process incur the validity page fault. Kernel handles the validity fault and the process will incur the protection fault if any one is present.

33. *In what way the protection fault handler concludes?*

After finishing the execution of the fault handler, it sets the *modify* and *protection* bits and clears the *copy on write* bit. It recalculates the process-priority and checks for signals.

33. *How the Kernel handles both the page stealer and the fault handler?*

The page stealer and the fault handler thrash because of the shortage of the memory. If the sum of the working sets of all processes is greater that the physical memory then the fault handler will usually sleep because it cannot allocate pages for a process. This results in the reduction of the system throughput because Kernel spends too much time in overhead, rearranging the memory in the frantic pace.

# RDBMS Concepts

## 1. What is database?

A database is a logically coherent collection of data with some inherent meaning, representing some aspect of real world and which is designed, built and populated with data for a specific purpose.

## 1. What is DBMS?

It is a collection of programs that enables user to create and maintain a database. In other words it is general-purpose software that provides the users with the processes of *defining*, *constructing* and *manipulating* the database for various applications.

## 1. What is a Database system?

The database and DBMS software together is called as Database system.

## 1. Advantages of DBMS?

- Ø Redundancy is controlled.
- Ø Unauthorised access is restricted.
- Ø Providing multiple user interfaces.
- Ø Enforcing integrity constraints.
- Ø Providing backup and recovery.

## 1. Disadvantage in File Processing System?

- Ø Data redundancy & inconsistency.
- Ø Difficult in accessing data.
- Ø Data isolation.
- Ø Data integrity.
- Ø Concurrent access is not possible.
- Ø Security Problems.

## 1. Describe the three levels of data abstraction?

The are three levels of abstraction:

- Ø *Physical level*: The lowest level of abstraction describes how data are stored.
- Ø *Logical level*: The next higher level of abstraction, describes what data are stored in database and what relationship among those data.
- Ø *View level*: The highest level of abstraction describes only part of entire database.

## 1. Define the "integrity rules"

There are two Integrity rules.

- Ø *Entity Integrity*: States that "Primary key cannot have NULL value"
- Ø *Referential Integrity*: States that "Foreign Key can be either a NULL value or should be Primary Key value of other relation.

## 1. What is extension and intension?

*Extension* - It is the number of tuples present in a table at any instance. This is time dependent.

*Intension* - It is a constant value that gives the name, structure of table and the constraints laid on it.

1. *What is System R? What are its two major subsystems?*

System R was designed and developed over a period of 1974-79 at IBM San Jose Research Center. It is a prototype and its purpose was to demonstrate that it is possible to build a Relational System that can be used in a real life environment to solve real life problems, with performance at least comparable to that of existing system.

Its two subsystems are

- Ø Research Storage
- Ø System Relational Data System.

1. *How is the data structure of System R different from the relational structure?*

Unlike Relational systems in System R

- Ø Domains are not supported
- Ø Enforcement of candidate key uniqueness is optional
- Ø Enforcement of entity integrity is optional
- Ø Referential integrity is not enforced

1. *What is Data Independence?*

Data independence means that “the application is independent of the storage structure and access strategy of data”. In other words, The ability to modify the schema definition in one level should not affect the schema definition in the next higher level.

Two types of Data Independence:

- Ø Physical Data Independence: Modification in physical level should not affect the logical level.
- Ø Logical Data Independence: Modification in logical level should affect the view level.

*NOTE: Logical Data Independence is more difficult to achieve*

1. *What is a view? How it is related to data independence?*

A view may be thought of as a virtual table, that is, a table that does not really exist in its own right but is instead derived from one or more underlying base table. In other words, there is no stored file that direct represents the view instead a definition of view is stored in data dictionary.

Growth and restructuring of base tables is not reflected in views. Thus the view can insulate users from the effects of restructuring and growth in the database. Hence accounts for logical data independence.

13. *What is Data Model?*

A collection of conceptual tools for describing data, data relationships data semantics and constraints.

14. *What is E-R model?*

This data model is based on real world that consists of basic objects called entities and of relationship among these objects. Entities are described in a database by a set of attributes.

14. *What is Object Oriented model?*

This model is based on collection of objects. An object contains values stored in instance variables with in the object. An object also contains bodies of code

that operate on the object. These bodies of code are called methods. Objects that contain same types of values and the same methods are grouped together into classes.

16. *What is an Entity?*

It is a 'thing' in the real world with an independent existence.

16. *What is an Entity type?*

It is a collection (set) of entities that have same attributes.

16. *What is an Entity set?*

It is a collection of all entities of particular entity type in the database.

16. *What is an Extension of entity type?*

The collections of entities of a particular entity type are grouped together into an entity set.

16. *What is Weak Entity set?*

An entity set may not have sufficient attributes to form a primary key, and its primary key compromises of its partial key and primary key of its parent entity, then it is said to be Weak Entity set.

16. *What is an attribute?*

It is a particular property, which describes the entity.

16. *What is a Relation Schema and a Relation?*

A relation Schema denoted by  $R(A_1, A_2, \dots, A_n)$  is made up of the relation name  $R$  and the list of attributes  $A_i$  that it contains. A relation is defined as a set of tuples. Let  $r$  be the relation which contains set tuples  $(t_1, t_2, t_3, \dots, t_n)$ . Each tuple is an ordered list of  $n$ -values  $t=(v_1, v_2, \dots, v_n)$ .

16. *What is degree of a Relation?*

It is the number of attribute of its relation schema.

16. *What is Relationship?*

It is an association among two or more entities.

16. *What is Relationship set?*

The collection (or set) of similar relationships.

16. *What is Relationship type?*

Relationship type defines a set of associations or a relationship set among a given set of entity types.

16. *What is degree of Relationship type?*

It is the number of entity type participating.

25. *What is DDL (Data Definition Language)?*

A data base schema is specifies by a set of definitions expressed by a special language called DDL.

25. *What is VDL (View Definition Language)?*

It specifies user views and their mappings to the conceptual schema.

25. *What is SDL (Storage Definition Language)?*

This language is to specify the internal schema. This language may specify the mapping between two schemas.

25. *What is Data Storage - Definition Language?*

The storage structures and access methods used by database system are specified by a set of definition in a special type of DDL called data storage-definition language.

25. *What is DML (Data Manipulation Language)?*

This language that enable user to access or manipulate data as organised by appropriate data model.

Ø *Procedural DML or Low level:* DML requires a user to specify what data are needed and how to get those data.

Ø *Non-Procedural DML or High level:* DML requires a user to specify what data are needed without specifying how to get those data.

31. *What is DML Compiler?*

It translates DML statements in a query language into low-level instruction that the query evaluation engine can understand.

31. *What is Query evaluation engine?*

It executes low-level instruction generated by compiler.

31. *What is DDL Interpreter?*

It interprets DDL statements and record them in tables containing metadata.

34. *What is Record-at-a-time?*

The Low level or Procedural DML can specify and retrieve each record from a set of records. This retrieve of a record is said to be Record-at-a-time.

34. *What is Set-at-a-time or Set-oriented?*

The High level or Non-procedural DML can specify and retrieve many records in a single DML statement. This retrieve of a record is said to be Set-at-a-time or Set-oriented.

34. *What is Relational Algebra?*

It is procedural query language. It consists of a set of operations that take one or two relations as input and produce a new relation.

34. *What is Relational Calculus?*

It is an applied predicate calculus specifically tailored for relational databases proposed by E.F. Codd. E.g. of languages based on it are DSL ALPHA, QUEL.

34. *How does Tuple-oriented relational calculus differ from domain-oriented relational calculus*

The tuple-oriented calculus uses a tuple variables i.e., variable whose only permitted values are tuples of that relation. E.g. QUEL  
 The domain-oriented calculus has domain variables i.e., variables that range over the underlying domains instead of over relation. E.g. ILL, DEDUCE.

34. *What is normalization?*

It is a process of analysing the given relation schemas based on their Functional Dependencies (FDs) and primary key to achieve the properties

- Ø Minimizing redundancy
- Ø Minimizing insertion, deletion and update anomalies.

34. *What is Functional Dependency?*

A Functional dependency is denoted by  $X \twoheadrightarrow Y$  between two sets of attributes X and Y that are subsets of R specifies a constraint on the possible tuple that can form a relation state r of R. The constraint is for any two tuples t1 and t2 in r if t1[X] = t2[X] then they have t1[Y] = t2[Y]. This means the value of X component of a tuple uniquely determines the value of component Y.

34. *When is a functional dependency F said to be minimal?*

- Ø Every dependency in F has a single attribute for its right hand side.
- Ø We cannot replace any dependency  $X \twoheadrightarrow A$  in F with a dependency  $Y \twoheadrightarrow A$  where Y is a proper subset of X and still have a set of dependency that is equivalent to F.
- Ø We cannot remove any dependency from F and still have set of dependency that is equivalent to F.

34. *What is Multivalued dependency?*

Multivalued dependency denoted by  $X \twoheadrightarrow Y$  specified on relation schema R, where X and Y are both subsets of R, specifies the following constraint on any relation r of R: if two tuples t1 and t2 exist in r such that t1[X] = t2[X] then t3 and t4 should also exist in r with the following properties

- Ø t3[X] = t4[X] = t1[X] = t2[X]
- Ø t3[Y] = t1[Y] and t4[Y] = t2[Y]
- Ø t3[Z] = t2[Z] and t4[Z] = t1[Z]  
 where [Z = (R - (X U Y)) ]

34. *What is Lossless join property?*

It guarantees that the spurious tuple generation does not occur with respect to relation schemas after decomposition.

34. *What is 1 NF (Normal Form)?*

The domain of attribute must include only atomic (simple, indivisible) values.

34. *What is Fully Functional dependency?*

It is based on concept of full functional dependency. A functional dependency  $X \twoheadrightarrow Y$  is full functional dependency if removal of any attribute A from X means that the dependency does not hold any more.

34. *What is 2NF?*

A relation schema R is in 2NF if it is in 1NF and every non-prime attribute A in R is fully functionally dependent on primary key.

34. What is 3NF?

A relation schema R is in 3NF if it is in 2NF and for every FD  $X \twoheadrightarrow A$  either of the following is true

Ø X is a Super-key of R.

Ø A is a prime attribute of R.

In other words, if every non prime attribute is non-transitively dependent on primary key.

34. What is BCNF (Boyce-Codd Normal Form)?

A relation schema R is in BCNF if it is in 3NF and satisfies an additional constraint that for every FD  $X \twoheadrightarrow A$ , X must be a candidate key.

34. What is 4NF?

A relation schema R is said to be in 4NF if for every Multivalued dependency  $X \twoheadrightarrow Y$  that holds over R, one of following is true

Ø X is subset or equal to (or)  $XY = R$ .

Ø X is a super key.

34. What is 5NF?

A Relation schema R is said to be 5NF if for every join dependency  $\{R_1, R_2, \dots, R_n\}$  that holds R, one the following is true

Ø  $R_i = R$  for some i.

Ø The join dependency is implied by the set of FD, over R in which the left side is key of R.

51. What is Domain-Key Normal Form?

A relation is said to be in DKNF if all constraints and dependencies that should hold on the the constraint can be enforced by simply enforcing the domain constraint and key constraint on the relation.

52. What are partial, alternate,, artificial, compound and natural key?

*Partial Key:*

It is a set of attributes that can uniquely identify weak entities and that are related to same owner entity. It is sometime called as Discriminator.

*Alternate Key:*

All Candidate Keys excluding the Primary Key are known as Alternate Keys.

*Artificial Key:*

If no obvious key, either stand alone or compound is available, then the last resort is to simply create a key, by assigning a unique number to each record or occurrence. Then this is known as developing an artificial key.

*Compound Key:*

If no single data element uniquely identifies occurrences within a construct, then combining multiple elements to create a unique identifier for the construct is known as creating a compound key.

*Natural Key:*

When one of the data elements stored within a construct is utilized as the primary key, then it is called the natural key.

52. What is indexing and what are the different kinds of indexing?

Indexing is a technique for determining how quickly specific data can be found. Types:

- Ø Binary search style indexing
- Ø B-Tree indexing
- Ø Inverted list indexing
- Ø Memory resident table
- Ø Table indexing

52. What is system catalog or catalog relation? How is better known as?

A RDBMS maintains a description of all the data that it contains, information about every relation and index that it contains. This information is stored in a collection of relations maintained by the system called metadata. It is also called data dictionary.

52. What is meant by query optimization?

The phase that identifies an efficient execution plan for evaluating a query that has the least estimated cost is referred to as query optimization.

52. What is join dependency and inclusion dependency?

*Join Dependency:*

A Join dependency is generalization of Multivalued dependency. A JD  $\{R_1, R_2, \dots, R_n\}$  is said to hold over a relation R if  $R_1, R_2, R_3, \dots, R_n$  is a lossless-join decomposition of R. There is no set of sound and complete inference rules for JD.

*Inclusion Dependency:*

An Inclusion Dependency is a statement of the form that some columns of a relation are contained in other columns. A foreign key constraint is an example of inclusion dependency.

52. What is durability in DBMS?

Once the DBMS informs the user that a transaction has successfully completed, its effects should persist even if the system crashes before all its changes are reflected on disk. This property is called durability.

52. What do you mean by atomicity and aggregation?

*Atomicity:*

Either all actions are carried out or none are. Users should not have to worry about the effect of incomplete transactions. DBMS ensures this by undoing the actions of incomplete transactions.

*Aggregation:*

A concept which is used to model a relationship between a collection of entities and relationships. It is used when we need to express a relationship among relationships.

52. What is a Phantom Deadlock?

In distributed deadlock detection, the delay in propagating local information might cause the deadlock detection algorithms to identify deadlocks that do not really exist. Such situations are called phantom deadlocks and they lead to unnecessary aborts.

52. *What is a checkpoint and When does it occur?*

A Checkpoint is like a snapshot of the DBMS state. By taking checkpoints, the DBMS can reduce the amount of work to be done during restart in the event of subsequent crashes.

52. *What are the different phases of transaction?*

Different phases are

- Ø Analysis phase
- Ø Redo Phase
- Ø Undo phase

52. *What do you mean by flat file database?*

It is a database in which there are no programs or user access languages. It has no cross-file capabilities but is user-friendly and provides user-interface management.

52. *What is "transparent DBMS"?*

It is one, which keeps its Physical Structure hidden from user.

52. *Brief theory of Network, Hierarchical schemas and their properties*

Network schema uses a graph data structure to organize records example for such a database management system is CTCG while a hierarchical schema uses a tree data structure example for such a system is IMS.

52. *What is a query?*

A query with respect to DBMS relates to user commands that are used to interact with a data base. The query language can be classified into data definition language and data manipulation language.

52. *What do you mean by Correlated subquery?*

Subqueries, or nested queries, are used to bring back a set of rows to be used by the parent query. Depending on how the subquery is written, it can be executed once for the parent query or it can be executed once for each row returned by the parent query. If the subquery is executed for each row of the parent, this is called a *correlated subquery*.

A correlated subquery can be easily identified if it contains any references to the parent subquery columns in its WHERE clause. Columns from the subquery cannot be referenced anywhere else in the parent query. The following example demonstrates a non-correlated subquery.

E.g. Select \* From CUST Where '10/03/1990' IN (Select ODATE From ORDER Where CUST.CNUM = ORDER.CNUM)

52. *What are the primitive operations common to all record management systems?*

Addition, deletion and modification.

52. *Name the buffer in which all the commands that are typed in are stored*

**'Edit'** Buffer

52. *What are the unary operations in Relational Algebra?*

PROJECTION and SELECTION.

52. Are the resulting relations of *PRODUCT* and *JOIN* operation the same?

No.

*PRODUCT*: Concatenation of every row in one relation with every row in another.

*JOIN*: Concatenation of rows from one relation and related rows from another.

52. What is *RDBMS KERNEL*?

Two important pieces of RDBMS architecture are the kernel, which is the software, and the data dictionary, which consists of the system-level data structures used by the kernel to manage the database

You might think of an RDBMS as an operating system (or set of subsystems), designed specifically for controlling data access; its primary functions are storing, retrieving, and securing data. An RDBMS maintains its own list of authorized users and their associated privileges; manages memory caches and paging; controls locking for concurrent resource usage; dispatches and schedules user requests; and manages space usage within its table-space structures

52. Name the sub-systems of a RDBMS

I/O, Security, Language Processing, Process Control, Storage Management, Logging and Recovery, Distribution Control, Transaction Control, Memory Management, Lock Management

52. Which part of the RDBMS takes care of the data dictionary? How

Data dictionary is a set of tables and database objects that is stored in a special area of the database and maintained exclusively by the kernel.

52. What is the job of the information stored in data-dictionary?

The information in the data dictionary validates the existence of the objects, provides access to them, and maps the actual physical storage location.

52. Not only RDBMS takes care of locating data it also \_\_\_\_\_  
determines an optimal access path to store or retrieve the data

52. How do you communicate with an RDBMS?

You communicate with an RDBMS using Structured Query Language (SQL)

52. Define SQL and state the differences between SQL and other conventional programming Languages

SQL is a nonprocedural language that is designed specifically for data access operations on normalized relational database structures. The primary difference between SQL and other conventional programming languages is that SQL statements specify what data operations should be performed rather than how to perform them.

52. Name the three major set of files on disk that compose a database in Oracle

There are three major sets of files on disk that compose a database. All the files are binary. These are

- Ø Database files
- Ø Control files
- Ø Redo logs

The most important of these are the database files where the actual data resides. The control files and the redo logs support the functioning of the architecture itself.

All three sets of files must be present, open, and available to Oracle for any data on the database to be useable. Without these files, you cannot access the database, and the database administrator might have to recover some or all of the database using a backup, if there is one.

#### 52. What is an Oracle Instance?

The Oracle system processes, also known as Oracle background processes, provide functions for the user processes—functions that would otherwise be done by the user processes themselves

Oracle database-wide system memory is known as the SGA, the *system global area* or *shared global area*. The data and control structures in the SGA are shareable, and all the Oracle background processes and user processes can use them.

The combination of the SGA and the Oracle background processes is known as an *Oracle instance*

#### 52. What are the four Oracle system processes that must always be up and running for the database to be useable

The four Oracle system processes that must always be up and running for the database to be useable include *DBWR* (Database Writer), *LGWR* (Log Writer), *SMON* (System Monitor), and *PMON* (Process Monitor).

#### 52. What are database files, control files and log files. How many of these files should a database have at least? Why?

##### *Database Files*

The database files hold the actual data and are typically the largest in size. Depending on their sizes, the tables (and other objects) for all the user accounts can go in one database file—but that's not an ideal situation because it does not make the database structure very flexible for controlling access to storage for different users, putting the database on different disk drives, or backing up and restoring just part of the database.

You must have at least one database file but usually, more than one files are used. In terms of accessing and using the data in the tables and other objects, the number (or location) of the files is immaterial.

The database files are fixed in size and never grow bigger than the size at which they were created

##### *Control Files*

The control files and redo logs support the rest of the architecture. Any database must have at least one control file, although you typically have more than one to guard against loss. The control file records the name of the database, the date and time it was created, the location of the database and redo logs, and the synchronization information to ensure that all three sets of files are always in step. Every time you add a new database or redo log file to the database, the information is recorded in the control files.

##### *Redo Logs*

Any database must have at least two redo logs. These are the journals for the database; the redo logs record all changes to the user objects or system objects. If any type of failure occurs, the changes recorded in the redo logs can be used to

bring the database to a consistent state without losing any committed transactions. In the case of non-data loss failure, Oracle can apply the information in the redo logs automatically without intervention from the DBA.

The redo log files are fixed in size and never grow dynamically from the size at which they were created.

52. *What is ROWID?*

The ROWID is a unique database-wide physical address for every row on every table. Once assigned (when the row is first inserted into the database), it never changes until the row is deleted or the table is dropped.

The ROWID consists of the following three components, the combination of which uniquely identifies the physical storage location of the row.

- Ø Oracle database file number, which contains the block with the rows
- Ø Oracle block address, which contains the row
- Ø The row within the block (because each block can hold many rows)

The ROWID is used internally in indexes as a quick means of retrieving rows with a particular key value. Application developers also use it in SQL statements as a quick way to access a row once they know the ROWID

52. *What is Oracle Block? Can two Oracle Blocks have the same address?*

Oracle "formats" the database files into a number of Oracle blocks when they are first created—making it easier for the RDBMS software to manage the files and easier to read data into the memory areas.

The block size should be a multiple of the operating system block size. Regardless of the block size, the entire block is not available for holding data; Oracle takes up some space to manage the contents of the block. This block header has a minimum size, but it can grow.

These Oracle blocks are the smallest unit of storage. Increasing the Oracle block size can improve performance, but it should be done only when the database is first created.

Each Oracle block is numbered sequentially for each database file starting at 1. Two blocks can have the same block address if they are in different database files.

52. *What is database Trigger?*

A database trigger is a PL/SQL block that can be defined to automatically execute for insert, update, and delete statements against a table. The trigger can be defined to execute once for the entire statement or once for every row that is inserted, updated, or deleted. For any one table, there are twelve events for which you can define database triggers. A database trigger can call database procedures that are also written in PL/SQL.

52. *Name two utilities that Oracle provides, which are used for backup and recovery.*

Along with the RDBMS software, Oracle provides two utilities that you can use to back up and restore the database. These utilities are *Export* and *Import*.

The *Export utility* dumps the definitions and data for the specified part of the database to an operating system binary file. The *Import utility* reads the file produced by an export, recreates the definitions of objects, and inserts the data

If Export and Import are used as a means of backing up and recovering the database, all the changes made to the database cannot be recovered since the export

was performed. The best you can do is recover the database to the time when the export was last performed.

52. *What are stored-procedures? And what are the advantages of using them.*

Stored procedures are database objects that perform a user defined operation. A stored procedure can have a set of compound SQL statements. A stored procedure executes the SQL commands and returns the result to the client. Stored procedures are used to reduce network traffic.

52. *How are exceptions handled in PL/SQL? Give some of the internal exceptions' name*

PL/SQL exception handling is a mechanism for dealing with run-time errors encountered during procedure execution. Use of this mechanism enables execution to continue if the error is not severe enough to cause procedure termination.

The exception handler must be defined within a subprogram specification. Errors cause the program to raise an exception with a transfer of control to the exception-handler block. After the exception handler executes, control returns to the block in which the handler was defined. If there are no more executable statements in the block, control returns to the caller.

#### *User-Defined Exceptions*

PL/SQL enables the user to define exception handlers in the declarations area of subprogram specifications. User accomplishes this by naming an exception as in the following example:

```
ot_failure EXCEPTION;
```

In this case, the exception name is `ot_failure`. Code associated with this handler is written in the EXCEPTION specification area as follows:

```
EXCEPTION
  when OT_FAILURE then
    out_status_code := g_out_status_code;
    out_msg         := g_out_msg;
```

The following is an example of a subprogram exception:

```
EXCEPTION
  when NO_DATA_FOUND then
    g_out_status_code := 'FAIL';
    RAISE ot_failure;
```

Within this exception is the RAISE statement that transfers control back to the `ot_failure` exception handler. This technique of raising the exception is used to invoke all user-defined exceptions.

#### *System-Defined Exceptions*

Exceptions internal to PL/SQL are raised automatically upon error. `NO_DATA_FOUND` is a system-defined exception. Table below gives a complete list of internal exceptions.

#### *PL/SQL internal exceptions.*

In addition to this list of exceptions, there is a catch-all exception named `OTHERS` that traps all errors for which specific error handling has not been established.

52. *Does PL/SQL support "overloading"? Explain*

The concept of *overloading* in PL/SQL relates to the idea that you can define procedures and functions with the same name. PL/SQL does not look only at the referenced name, however, to resolve a procedure or function call. The count and data types of formal parameters are also considered.

PL/SQL also attempts to resolve any procedure or function calls in locally defined packages before looking at globally defined packages or internal functions. To further ensure calling the proper procedure, you can use the dot notation. Prefacing a procedure or function name with the package name fully qualifies any procedure or function reference.

52. *Tables derived from the ERD*

- a) Are totally unnormalised
- b) Are always in 1NF
- c) Can be further denormalised
- d) May have multi-valued attributes

(b) Are always in 1NF

52. *Spurious tuples may occur due to*

- i. *Bad normalization*
- ii. *Theta joins*
- iii. *Updating tables from join*

- a) i & ii
- b) ii & iii
- c) i & iii
- d) ii & iii

(a) i & iii because theta joins are joins made on keys that are not primary keys.

52. *A B C is a set of attributes. The functional dependency is as follows*

$AB \rightarrow B$

$AC \rightarrow C$

$C \rightarrow B$

- a) is in 1NF
- b) is in 2NF
- c) is in 3NF
- d) is in BCNF

(a) is in 1NF since  $(AC)^+ = \{ A, B, C \}$  hence AC is the primary key. Since  $C \rightarrow B$  is a FD given, where neither C is a Key nor B is a prime attribute, this it is not in 3NF. Further B is not functionally dependent on key AC thus it is not in 2NF. Thus the given FDs is in 1NF.

52. *In mapping of ERD to DFD*

- a) entities in ERD should correspond to an existing entity/store in DFD
- b) entity in DFD is converted to attributes of an entity in ERD
- c) relations in ERD has 1 to 1 correspondence to processes in DFD
- d) relationships in ERD has 1 to 1 correspondence to flows in DFD

(a) entities in ERD should correspond to an existing entity/store in DFD

52. A dominant entity is the entity
- a) on the N side in a 1 : N relationship
  - b) on the 1 side in a 1 : N relationship
  - c) on either side in a 1 : 1 relationship
  - d) nothing to do with 1 : 1 or 1 : N relationship

(b) on the 1 side in a 1 : N relationship

52. Select 'NORTH', CUSTOMER From CUST\_DTLS Where REGION = 'N' Order By  
CUSTOMER Union Select 'EAST', CUSTOMER From CUST\_DTLS Where  
REGION = 'E' Order By CUSTOMER

The above is

- a) Not an error
- b) Error - the string in single quotes 'NORTH' and 'SOUTH'
- c) Error - the string should be in double quotes
- d) Error - ORDER BY clause

(d) Error - the ORDER BY clause. Since ORDER BY clause cannot be used in UNIONS

52. What is Storage Manager?

It is a program module that provides the interface between the low-level data stored in database, application programs and queries submitted to the system.

52. What is Buffer Manager?

It is a program module, which is responsible for fetching data from disk storage into main memory and deciding what data to be cache in memory.

52. What is Transaction Manager?

It is a program module, which ensures that database, remains in a consistent state despite system failures and concurrent transaction execution proceeds without conflicting.

52. What is File Manager?

It is a program module, which manages the allocation of space on disk storage and data structure used to represent information stored on a disk.

52. What is Authorization and Integrity manager?

It is the program module, which tests for the satisfaction of integrity constraint and checks the authority of user to access data.

52. What are stand-alone procedures?

Procedures that are not part of a package are known as stand-alone because they independently defined. A good example of a stand-alone procedure is one written in a SQL\*Forms application. These types of procedures are not available for reference from other Oracle tools. Another limitation of stand-alone procedures is that they are compiled at run time, which slows execution.

52. What are cursors give different types of cursors.

PL/SQL uses cursors for all database information accesses statements. The language supports the use two types of cursors

- Ø Implicit
- Ø Explicit

52. What is cold backup and hot backup (in case of Oracle)?

Ø Cold Backup:

It is copying the three sets of files (database files, redo logs, and control file) when the instance is shut down. This is a straight file copy, usually from the disk directly to tape. You must shut down the instance to guarantee a consistent copy.

If a cold backup is performed, the only option available in the event of data file loss is restoring all the files from the latest backup. All work performed on the database since the last backup is lost.

Ø Hot Backup:

Some sites (such as worldwide airline reservations systems) cannot shut down the database while making a backup copy of the files. The cold backup is not an available option.

So different means of backing up database must be used — the hot backup. Issue a SQL command to indicate to Oracle, on a tablespace-by-tablespace basis, that the files of the tablespace are to be backed up. The users can continue to make full use of the files, including making changes to the data. Once the user has indicated that he/she wants to back up the tablespace files, he/she can use the operating system to copy those files to the desired backup destination.

The database must be running in ARCHIVELOG mode for the hot backup option.

If a data loss failure does occur, the lost database files can be restored using the hot backup and the online and offline redo logs created since the backup was done. The database is restored to the most consistent state without any loss of committed transactions.

52. What are Armstrong rules? How do we say that they are complete and/or sound

The well-known inference rules for FDs

Ø Reflexive rule :

If  $Y$  is subset or equal to  $X$  then  $X \twoheadrightarrow Y$ .

Ø Augmentation rule:

If  $X \twoheadrightarrow Y$  then  $XZ \twoheadrightarrow YZ$ .

Ø Transitive rule:

If  $\{X \twoheadrightarrow Y, Y \twoheadrightarrow Z\}$  then  $X \twoheadrightarrow Z$ .

Ø Decomposition rule :

If  $X \twoheadrightarrow YZ$  then  $X \twoheadrightarrow Y$ .

Ø Union or Additive rule:

If  $\{X \twoheadrightarrow Y, X \twoheadrightarrow Z\}$  then  $X \twoheadrightarrow YZ$ .

Ø Pseudo Transitive rule :

If  $\{X \twoheadrightarrow Y, WY \twoheadrightarrow Z\}$  then  $WX \twoheadrightarrow Z$ .

Of these the first three are known as Armstrong Rules. They are sound because it is enough if a set of FDs satisfy these three. They are called complete because using these three rules we can generate the rest all inference rules.

52. How can you find the minimal key of relational schema?

Minimal key is one which can identify each tuple of the given relation schema uniquely. For finding the minimal key it is required to find the closure that is the set of all attributes that are dependent on any given set of attributes under the given set of functional dependency.

**Algo. I** Determining  $X^+$ , closure for X, given set of FDs F

1. Set  $X^+ = X$
1. Set Old  $X^+ = X^+$
1. For each FD  $Y \twoheadrightarrow Z$  in F and if Y belongs to  $X^+$  then add Z to  $X^+$
1. Repeat steps 2 and 3 until Old  $X^+ = X^+$

**Algo. II** Determining minimal K for relation schema R, given set of FDs F

1. Set K to R that is make K a set of all attributes in R
1. For each attribute A in K
  - a. Compute  $(K - A)^+$  with respect to F
  - a. If  $(K - A)^+ = R$  then set  $K = (K - A)^+$

52. What do you understand by dependency preservation?

Given a relation R and a set of FDs F, dependency preservation states that the closure of the union of the projection of F on each decomposed relation Ri is equal to the closure of F. i.e.,

$$((\Pi_{R_1}(F)) \cup \dots \cup (\Pi_{R_n}(F)))^+ = F^+$$

if decomposition is not dependency preserving, then some dependency is lost in the decomposition.

52. What is meant by Proactive, Retroactive and Simultaneous Update.

*Proactive Update:*

The updates that are applied to database before it becomes effective in real world .

*Retroactive Update:*

The updates that are applied to database after it becomes effective in real world .

*Simultaneous Update:*

The updates that are applied to database at the same time when it becomes effective in real world .

52. What are the different types of JOIN operations?

*Equi Join:* This is the most common type of join which involves only equality comparisons. The disadvantage in this type of join is that there.