1. What's new in Sybase ASE 15?

Ans:

- Prior to Adaptive Server 15.0, all indexes were global. With Adaptive Server 15.0, you can create local as well as global indexes.
  Read more on: http://sybaserays.com/local-and-global-indexes-on-partitioned-tables
- Scrollable cursor
  Read more on: http://sybaserays.com/scrollable-cursors-2
- datachange() function: The datachange function is the key to identifying whether update statistics operations on a table, index, partition, or column is neces - sary. The datachange function returns a value to indicate how much the data has changed within a table, partition, index, or column. A value of 0% indicates no changes to the object are measured by the datachange function. As the data changes, the value returned will increase.
- set option show_missing_stats
- Optimization goals: The new "optimization goal" configuration parameter allows the user to choose the optimization strategy that best fits the query environment.
- Join types: HASH JOIN, MERGE JOIN, NESTED LOOP JOIN, NARY NESTED LOOP JOIN
- New MDA table monOpenPartitionActivity: A new MDA table for ASE 15. This table is very similar to the monOpenObjectActivity table but at the partition level.
- Computed columns.

2. Explain Isolation levels.

Ans:

Isolation level: The ANSI SQL standard defines four levels of isolation for transactions. Each isolation level specifies the kinds of actions that are not permitted while concurrent transactions are executing. Higher levels include the restrictions imposed by the lower levels:

Level 0: ensures that data written by one transaction represents the actual data. It prevents other transactions from changing data that has already been modified (through an insert, delete, update, and so on) by an uncommitted transaction. The other transactions are blocked from modifying that data until the transaction commits. However, other transactions can still read the uncommitted data, which results in dirty reads.

Level 1: prevents dirty reads. Such reads occur when one transaction modifies a row, and a second transaction reads that row before the first transaction commits the change. If the first transaction rolls back the change, the information read by the second transaction becomes invalid. This is the default isolation level supported by Adaptive Server.

Level 2: prevents nonrepeatable reads. Such reads occur when one transaction reads a row and a second transaction modifies that row. If the second transaction commits its change, subsequent reads by the first transaction yield different results than the original read.
Adaptive Server supports this level for data-only-locked tables. It is not supported for allpages-locked tables.

Level 3: ensures that data read by one transaction is valid until the end of that transaction, preventing phantom rows. Adaptive Server supports this level through the holdlock keyword of the select statement, which applies a read-lock on the specified data. Phantom rows occur when one transaction reads a set of rows that satisfy a search condition, and then a second transaction

modifies the data (through an insert, delete, update, and so on). If the first transaction repeats the read with the same search conditions, it obtains a different set of rows.

3. Why do we require locking?

Ans:

Locking is a concurrency control mechanism: it ensures the consistency of data within and across transactions. Locking is needed in a multiuser environment, since several users may be working with the same data at the same time.

4. What are the locking schemes available in Sybase ASE?

Ans:

Locking	Description
Allpages	locks datapages and index pages
Datapages	which locks only the data pages
Datarows	which locks only the data rows

5. Please explain each locking scheme available in Sybase ASE?

Ans:

Allpages locking: Allpages locking locks both data pages and index pages. When a query updates a value in a row in an allpages-locked table, the data page is locked with an exclusive lock. Any index pages affected by the update are also locked with exclusive locks. These locks are transactional, meaning that they are held until the end of the transaction.

In many cases, the concurrency problems that result from allpages locking arise from the index page locks, rather than the locks on the data pages themselves. Data pages have longer rows than indexes, and often have a small number of rows per page. If index keys are short, an index page can store between 100 and 200 keys. An exclusive lock on an index page can block other users who need to access any of the rows referenced by the index page, a far greater number of rows than on a locked data page.

Datapages locking: In datapages locking, entire data pages are still locked, but index pages are not locked. When a row needs to be changed on a data page, that page is locked, and the lock is held until the end of the transaction. The updates to the index pages are performed using latches, which are non transactional. Latches are held only as long as required to perform the physical changes to the page and are then released immediately. Index page entries are implicitly locked by locking the data page. No transactional locks are held on index pages.

Datarows locking: In datarows locking, row-level locks are acquired on individual rows on data pages. Index rows and pages are not locked. When a row needs to be changed on a data page, a non-transactional latch is acquired on the page. The latch is held while the physical change is made to the data page, and then the latch is released. The lock on the data row is held until the end of the transaction. The index rows are updated, using latches on the index page, but are not locked. Index entries are implicitly locked by acquiring a lock on the data row.

Note: Page or row locks are less restrictive (or smaller) than table locks. A page lock locks all the rows on data page or an index page; a table lock locks an entire table. A row lock locks only a single row on a page. Adaptive Server uses page or row locks whenever possible to reduce contention and to improve concurrency.

Note: Adaptive Server uses a table lock to provide more efficient locking when an entire table or a large number of pages or rows will be accessed by a statement. Locking strategy is directly tied to

the query plan, so the query plan can be as important for its locking strategies as for its I/O implications.

6. What are the types of page and row locks? Please explain each lock type.

Ans:

Shared locks, Exclusive locks, update locks.

Shared lock: Adaptive Server applies shared locks for read operations. If a shared lock has been applied to a data page or data row or to an index page, other transactions can also acquire a shared lock, even when the first transaction is active. However, no transaction can acquire an exclusive lock on the page or row until all shared locks on the page or row are released. This means that many transactions can simultaneously read the page or row, but no transaction can change data on the page or row while a shared lock exists. Transactions that need an exclusive lock wait or "block" for the release of the shared locks before continuing.

By default, Adaptive Server releases shared locks after it finishes scanning the page or row. It does not hold shared locks until the statement is completed or until the end of the transaction unless requested to do so by the user.

Exclusive lock: Adaptive Server applies an exclusive lock for a data modification operation. When a transaction gets an exclusive lock, other transactions cannot acquire a lock of any kind on the page or row until the exclusive lock is released at the end of its transaction. The other transactions wait or "block" until the exclusive lock is released.

Update lock: Adaptive Server applies an update lock during the initial phase of an update, delete, or fetch (for cursors declared for update) operation while the page or row is being read. The update lock allows shared locks on the page or row, but does not allow other update or exclusive locks.

Note: Update locks help avoid deadlocks and lock contention.

Note: In general, read operations acquire shared locks, and write operations acquire exclusive locks. For operations that delete or update data, Adaptive Server applies page-level or row-level exclusive and update locks only if the column used in the search argument is part of an index. If no index exists on any of the search arguments, Adaptive Server must acquire a table-level lock.

Examples: Page locks and row locks

Statement	All pages locked table	Datarows locked table
select balance from account where acct_number = 25	Shared page lock	Shared row lock
insert account values (34, 500)	Exclusive page lock on data page and exclusive page lock on leaflevel index page	Exclusive row lock
delete account where acct_number = 25	Update page locks followed by exclusive page locks on data pages and exclusive page locks on leaflevel index pages	Update row locks followed by exclusive row locks on each affected row
update account set balance = 0 where acct_number = 25	Update page lock on data page and exclusive page lock on data page	Update row lock followed by exclusive row lock

7. What are the types of Table locks? Please explain each lock type.

Ans: Intent locks, Shared lock, Exclusive lock

Intent lock: An intent lock indicates that page-level or row-level locks are currently held on a table. Adaptive Server applies an intent table lock with each shared or exclusive page or row lock, so an intent lock can be either an exclusive lock or a shared lock. Setting an intent lock prevents other transactions from subsequently acquiring conflicting table-level locks on the table that contains that locked page. An intent lock is held as long as page or row locks are in effect for the transaction.

Shared lock: This lock is similar to a shared page or lock, except that it affects the entire table. Adaptive Server applies a shared table lock for a select command with a holdlock clause if the command does not use an index.

Note: A create nonclustered index command also acquires a shared table lock.

Exclusive lock: This lock is similar to an exclusive page or row lock, except it affects the entire table. update and delete statements require exclusive table locks if their search arguments do not reference indexed columns of the object.

Note: Adaptive Server applies an exclusive table lock during a create clustered index command.

8. What is demand lock?

Ans:

Adaptive Server sets a demand lock to indicate that a transaction is next in the queue to lock a table, page, or row. Since many readers can hold shared locks on a given page, row, or table, tasks that require exclusive locks are queued after a task that already holds a shared lock. Adaptive Server allows up to three readers' tasks to skip over a queued update task.

After a write transaction has been skipped over by three tasks or families (in the case of queries running in parallel) that acquire shared locks, Adaptive Server gives a demand lock to the write transaction.

As soon as the readers queued ahead of the demand lock release their locks, the write transaction acquires its lock and is allowed to proceed. The read transactions queued behind the demand lock wait for the write transaction to finish and release its exclusive lock.

9. Explain use of latches in locking process?

Ans:

Latches are non-transactional synchronization mechanisms used to guarantee the physical consistency of a page. While rows are being inserted, updated or deleted, only one Adaptive Server process can have access to the page at the same time. Latches are used for datapages and datarows locking but not for allpages locking.

The most important distinction between a lock and a latch is the duration:

A lock can persist for a long period of time: while a page is being scanned, while a disk read or network write takes place, for the duration of a statement, or for the duration of a transaction.

A latch is held only for the time required to insert or move a few bytes on a data page, to copy pointers, columns or rows, or to acquire a latch on another index page.

10. What is lock contention?

Ans:

Lock contention: Locking affects performance when one process holds locks that prevent another process from accessing needed data. The process that is blocked by the lock sleeps until the lock is released. This is called lock contention.

11. How do you specify locking scheme while creating a table?
Ans:
create table table_name (column_name_list)
[lock {datarows | datapages | allpages}]
Example:
create table employee_new
(emp_id char(4) not null,
emp_name varchar(50) null,
status int not null)
lock datarows

12. How will you specify a locking scheme while creating a table using 'Select into'?
Ans:
Syntax:
select [all | distinct] select_list
into [[database.]owner.]table_name
lock {datarows | datapages | allpages}
from ...
Example:
*select * into employee_test*
lock allpages
from employee_new

13. What will be the locking scheme of a table(employee_test), If you do not specify a locking scheme with 'select into' command while for creating a table (employee_test)?
*select * into employee_test*
from employee_new

Ans: If you do not specify a locking scheme with select into, the new table uses the server-wide default locking scheme, as defined by the configuration parameter lock scheme.

14. How can you keep track of current state of transaction.?
Ans: The global variable @@transtate keeps track of the current state of a transaction. It may contain following values.

Value	Description
0	Transaction in progress. A transaction is in effect; the previous statement executed successfully.
1	Transaction succeeded. The transaction completed and committed its changes.
2	Statement aborted. The previous statement was aborted; no effect on the transaction.
3	Transaction aborted. The transaction aborted and rolled back any changes.

Note: Adaptive Server does not clear @@transtate after every statement. It changes @@transtate only in response to an action taken by a transaction. Syntax and compile errors do not affect the value of @@transtate

15. What is save point?
Ans: A savepoint is a marker that a user puts inside a transaction to indicate a point to which it can be rolled back. You can commit only certain portions of a batch by rolling back the undesired portion to a savepoint before committing the entire batch.

16. Explain nesting transactions?
Ans:
You can nest transactions within other transactions. When you nest begin transaction and commit transaction statements, the **outermost pair** actually **begin** and **commit** the transaction. The inner pairs just keep track of the nesting level. Adaptive Server does not commit the transaction until the commit transaction that matches the outermost begin transaction is issued. Normally, this transaction "nesting" occurs as stored procedures or triggers that contain begin/commit pairs call each other.
The @@trancount global variable keeps track of the current nesting level for transactions. An initial implicit or explicit begin transaction sets @@trancount to 1. Each subsequent begin transaction increments @@trancount, and a commit transaction decrements it. Firing a trigger also increments @@trancount, and the transaction begins with the statement that causes the trigger to fire. Nested transactions are not committed unless @@trancount equals 0.

17. What is transaction log?
Ans:
Every change to a database, whether it is the result of a single update statement or a grouped set of SQL statements, is recorded in the system table **syslogs**. This table is called the transaction log.

18. What are the transaction modes are supported by Sybase ASE?
Ans:
Adaptive Server supports following transaction modes:
Chained mode and Unchained mode
Chained mode: Implicitly begins a transaction before any data retrieval or modification statement. These statements include: delete, insert, open, fetch, select, and update. You must still explicitly end the transaction with commit transaction or rollback transaction.
Example:
set chained on
insert into employee
values (106, 'Steve', '01 Jan 1973', 1)

begin transaction
delete from employee where emp_id = 106

rollback transaction

*select * from employee*

emp_id	emp_name	dob	status
102	Mike	Nov 25 1979 12:00AM	1
103	Salman	Jan 1 1980 12:00AM	0
101	Abhisek	Jun 25 1983 12:00AM	1
104	Steve	Jan 1 1973 12:00AM	1

In above example, ASE implicitly executes begin transaction statement just before the insert command and rollback whole transaction (including insert statement).

Unchained mode: The default mode. Requires explicit begin transaction statements paired with commit transaction or rollback transaction statements to complete the transaction.
Example: The rollback affects only the delete statement, so employee still contains the inserted row.
insert into employee
values (105, 'Thomas', '02 Jan 1973', 1)

begin transaction

delete from employee where emp_id = 105

rollback transaction

*select * from employee*

emp_id	emp_name	dob	status
102	Mike	Nov 25 1979 12:00AM	1
103	Salman	Jan 1 1980 12:00AM	0
101	Abhisek	Jun 25 1983 12:00AM	1
104	Steve	Jan 1 1973 12:00AM	1
105	Thomas	Jan 2 1973 12:00AM	1

19. How can you change isolation level for a query?

Ans:

You can change the isolation level for a query by using the at isolation clause with the select or readtext statements. The at isolation clause supports isolation levels 0 (read uncommitted), 1(read committed), and 3 (serializable). It does not support isolation level 2.

Example:

*select **
from titles
at isolation read uncommitted

or

*select **
from titles
at isolation 0

20. If a trigger fires in a transaction at level 0, Trigger's which isolation level will be set by Adaptive Server before executing trigger's first statement?

Ans: 1

Note: Since triggers are fired by data modification statements (like insert), all triggers execute at either the transaction's isolation level or isolation level 1, whichever is higher. So, if a trigger fires in a transaction at level 0, Adaptive Server sets the trigger's isolation level to 1 before executing its first statement.

21. Describe how ASE resolves a deadlock?

Ans:

Adaptive Server checks for deadlocks and chooses the user whose transaction has accumulated the least amount of CPU time as the victim. Adaptive Server rolls back that user's transaction, notifies the application program of this action with message number 1205, and allows the other process to move forward.

22. What does update statistics command do?

Ans:

The update statistics commands create statistics, if there are no statistics for a particular column, or replaces existing statistics if they already exist. The statistics are stored in the system tables systabstats and sysstatistics.

Syntax:

For Table:

update statistics table_name

[[index_name] | [(column_list)]]
[using step values]
[with consumers = consumers]

For Index:

update index statistics table_name [index_name]
[using step values]
[with consumers = consumers]

For All (Table and Index):

update all statistics table_name

The effects of the commands and their parameters are:

For update statistics	
table_name	Generates statistics for the leading column in each index on the table.
table_name index_name	Generates statistics for all columns of the index.
table_name (column_name)	Generates statistics for only this column.
table_name (column_name, column_name...)	Generates a **histogram** for the leading column in the set, and multi column **density values** for the prefix subsets.
For update index statistics	
table_name	Generates statistics for all columns in all indexes on the table.
table_name index_name	Generates statistics for all columns in this index.
For update all statistics:	
table_name	Generates statistics for all columns of a table.

23. What is use of table partition? And What are the partition types available in Sybase ASE 15?
Ans:

- Improved scalability.
- Improved performance – concurrent multiple I/O on different partitions, and multiple threads on multiple CPUs working concurrently on multiple partitions.
- Faster response time.
- Partition transparency to applications.
- Very large database (VLDB) support – concurrent scanning of multiple partitions of very large tables.
- Range partitioning to manage historical data

Partition types available in Sybase ASE 15?

- Range partitioning
- Hash partitioning
- List partitioning
- Round-robin partitioning

Read more on: http://sybaserays.com/partitioning-strategies

24. How will you explicitly insert data into an IDENTITY column of a table?
Ans:
set identity_insert <table_name> on
Example:
set identity_insert sales_daily on

And insert value for identity column as well in insert statement.
Example:
Insert sales_daily (syb_identity, stor_id) values (101, "1349")

25. Explain direct updates and deferred updates?
Ans:
Direct updates: Adaptive Server performs direct updates in a single pass:

- It locates the affected index and data rows.
- It writes the log records for the changes to the transaction log.
- It makes the changes to the data pages and any affected index pages.

Deferred updates: Adaptive Server uses deferred updates when direct update conditions are not met. A deferred update is the slowest type of update. In a deferred update, Adaptive Server:

- Locates the affected data rows, writing the log records for deferred delete and insert of the data pages as rows are located.
- Reads the log records for the transaction and performs the deletes on the data pages and any affected index rows.
- Reads the log records a second time, and performs all inserts on the data pages, and inserts any affected index rows.

26. Which update type is good (deferred or direct) and why?
Ans:
Direct update is always good.
Advantage of Direct update over the Deferred update:

- Direct updates require less overhead than deferred updates.
- Generally faster, as they limit the number of log scans, reduce logging.
- Save traversal of index B-trees (reducing lock contention), and save I/O because Adaptive Server does not have to refetch pages to perform modifications based on log records.

27. What would you suggest your team member to avoid deferred updates in your application?
Ans: I will recommend following things which can help to avoid deferred updates

- Create at least one unique index on the table to encourage more direct updates.
- Whenever possible, use nonkey columns in the where clause when updating a different key.
- If you do not use null values in your columns, declare them as not null in your create table statement.

28. What is convered query?
Ans: A query is covered if all the columns it uses come from one or more indexes. These columns include the columns you want the query to return as well as columns in any JOIN, WHERE, HAVING, and ORDER BY clause.

29. What is use of bcp command?
Ans:

- To import data from a file into a database table.
- To export data to a file from a database table.

30. What is the main difference between slow bcp and fast bcp?
Ans:
Slow bcp: logs each row insert that it makes, used for tables that have one or more indexes or triggers.
Fast bcp: logs only page allocations, copying data into tables without indexes or triggers at the fastest speed possible.

31. Does bcp utility fire any trigger that exists on a table in which you are importing a file?
Ans: No

32. What is heap table?
Ans: If you create a table on Adaptive Server, but do not create a clustered index, the table is stored as a heap.

33. Explain Procedure cache and explain how stored procedures are processed?
Ans:
Adaptive Server maintains an MRU/LRU (most recently used/least recently used) chain of stored procedure query plans. As users execute stored procedures, Adaptive Server looks in the procedure cache for a query plan to use. If a query plan is available, it is placed on the MRU end of the chain, and execution begins.
If no plan is in memory, or if all copies are in use, the query tree for the procedure is read from the **sysprocedures** table. It is then optimized, using the parameters provided to the procedure, and put on the MRU end of the chain, and execution begins. Plans at the LRU end of the page chain that are not in use are aged out of the cache.
The memory allocated for the procedure cache holds the optimized query plans (and occasionally trees) for all batches, including any triggers.
If more than one user uses a procedure or trigger simultaneously, there will be multiple copies of it in cache. If the procedure cache is too small, a user trying to execute stored procedures or queries that fire triggers receives an error message and must resubmit the query. Space becomes available when unused plans age out of the cache.

34. What is the difference between return status and return parameters of a stored procedure?
Ans:
Return status: indicates whether or not the stored procedure completed successfully.
Return parameters: report the parameter values back to the caller, who can then use conditional statements to check the returned value.

35. What is difference between rollback trigger and rollback transaction?
Ans: You can roll back triggers using either the rollback trigger statement or the rollback transaction statement (if the trigger is fired as part of a transaction). However, rollback trigger rolls back only the effect of the trigger and the statement that caused the trigger to fire; rollback transaction rolls back the entire transaction.
Example:
begin tran
insert into publishers (pub_id) values ("9999")
insert into publishers (pub_id) values ("9998")
commit tran
If the second insert statement causes a trigger on publishers to issue a **rollback trigger**, only that insert is affected; the first insert is not rolled back.
If that trigger issues a **rollback transaction** instead, both insert statements are rolled back as part of the transaction.

36. What is use of 'if update' clause in trigger?
Ans:
if update clause tests for an insert or update to a specified column (not for delete). For updates, if update clause evaluates to true when the column name is included in the set clause of an update statement, even if the update does not change the value of the column.

Example: Following trigger prevents updates to f_name column of employee table

```
create trigger Trg_Stop_update_fname
on employee
for update
as
if update (f_name)
begin
rollback transaction
print "We do not allow changes on column f_name"
end
```

if you execute below script, it will not update column 'f_name' and will print message "We do not allow changes on column f_name"

```
update employee set f_name ='test' where id = 1
```

Bleow update will update column without printing any message

```
update employee set l_name ='test' where id = 1
```

37. What are the advantages of using triggers?

Ans:

- Triggers can reduce network traffic.
- Triggers can cascade changes through related tables in the database.
- Triggers can disallow, or roll back, changes that would violate referential integrity, canceling the attempted data modification transaction.
- Triggers can enforce restrictions that are much more complex than those that are defined with rules.
- Triggers can perform simple "what if" analyses. For example, a trigger can compare the state of a table before and after a data modification and take action based on that comparison.

38. Does truncate table command fire a trigger of that table?

Ans: No

39. What is Optimization goal?

Ans:

The new "optimization goal" configuration parameter allows the user to choose the optimization strategy that best fits the query environment.

Optimization goals are a convenient way to match query demands with the best optimization techniques, thus ensuring optimal use of the optimizer's time and resources. The query optimizer allows you to configure three types of optimization goals, which you can specify at three tiers: server level, session level, and query level.

You can set the optimization goal at the desired level. The server-level optimization goal is overridden at the session level, which is overridden at the query level.

These optimization goals allow you to choose an optimization strategy that best fits your query environment:

parameter	Description
allrows_mix	The **default goal**, and the most useful goal in a mixed-query environment. allrows_mix balances the needs of **OLTP** and **DSS query environments**. allrows_mix is basically allrows_oltp + merge joins + parallelism.
allrows_dss	The most useful goal for **operational DSS queries of medium to high complexity**. Currently, this goal is provided on an experimental basis.
allrows_oltp	The optimizer considers only **nested-loop joins.** This option is basically allrows_mix + hash joins.

At the server level, use sp_configure. For example:
sp_configure "optimization goal", 0, "allrows_mix"
or
Modify the optimization goal configuration parameter in the Adaptive Server configuration file

At the session level, use set plan optgoal. For example:
set plan optgoal allrows_dss

At the query level, use a select or other DML command. For example:
*select * from A order by A.col_name1 plan*
"(use optgoal allrows_dss)"

40. What is scrollable cursor?
Ans:
"Scrollable" means that you can scroll through the cursor result set by fetching any, or many, rows, rather than one row at a time; you can also scan the result set repeatedly.

A scrollable cursor allows you to set the position of the cursor anywhere in the cursor result set for as long as the cursor is open, by specifying the option first, last, absolute, next, prior, absolute or relative in a fetch statement.
Syntax:
declare cursor_name
[cursor_sensitivity]
[cursor_scrollability] cursor
for cursor_specification
Read more on: http://sybaserays.com/scrollable-cursors-2/

41. What do you understand by Sensitivity of a cursor?
Ans:
Cursor (from ASE 15.0) can be either semi-sensitive or insensitive.

- **Insensitive:** The cursor shows only the result set as it is when the cursor is opened; data changes in the underlying tables are not visible
- **Semi-sensitive:** some changes in the base tables made since opening the cursor may appear in the result set. Data changes may or may not be visible to the semi-sensitive cursor.

Read more on: http://sybaserays.com/scrollable-cursors-2/

42. What type of performance issues (database) have you solved/worked in you projects?
Ans: By asking this question, Interviewer just wants to know that on what type of performance tuning issues you usually work in your current project and what was the complexity of the issues.

Performance tuning is a big topic. I would recommend to start with issues you have really worked and solved in your projects because you will be more confident in explaining those issues and can answer any counter question asked by Interviewer.

In general following things can be checked/action in case of performance issue (if any process is slow).

- Check tempdb usage
- Check transaction log
- Check query plan:

  if you include set noexec, you can view the query plan without running the query.

  set noexec on
  go
  To see query plans, use:
  set showplan on
  To stop displaying query plans, use:
  set showplan off
  Example:
  set noexec on
  go
  set showplan on
  go
  *select * from really_big_table*
  *select * from really_really_big_table*
  go
- Check Indexes: check if proper indexes are being used or not. You can check this in query plan displayed by above step (showplan).
- Checks when you ran last update ***statistics*** *(for tables belong to slow process).* If you did not run update statistics since long time then please update.

43. There are two indexes idx1, idx2 on a table '*employee_transactions*'. One SQL query always uses index idx2 instead of idx1 but you think if query use idx1 then it will be be fast. What would you do so ASE uses idx1 instead of idx2?

Ans: We can force index on a table.

Example:

*select **
from 'employee_transactions et, employee t (index idx1)
where e.emp_id = et.emp_id
and …..

44. How can you get total rows of a table without using query *'select count(*) from table_name'*? (count query should not be used)

Ans:

To get a row count indirectly, without querying the table, use the **sp_spaceused** system stored procedure. Keep in mind that sp_spaceused gives an estimate based on an average number of rows per page, and it is more accurate if you run update statistics or dbcc checktable first.

Syntax:

sp_spaceused table_name

Note: The row count is returned in the column **rowtotal**.

45. What you do when a segment gets full?
Ans:
Wrong, a segment can never get full (even though some error messages state something to that extent). A segment is a "label" for one or more database device fragments; the fragments to which that label has been mapped can get full, but the segments themselves cannot. (Well, Ok, this is a bit of trick question… when those device fragments full up; you either add more space, or clean up old / redundant data.)
46. What is sp_recompile?
Ans:
sp_recompile causes each stored procedure and trigger that uses the named table to be recompiles the next time it runs.
Usage: The queries used by stored procedure and triggers are optimized only once, when they are compiled. As you add indexes or make other changes to your database that affect its statistics, your compiled stored procedures and triggers may lose efficiency. By recompiling the stored procedures and triggers that act on a table, you can optimize the queries for maximum efficiency.

47. How to lock a table in sybase? give me its text command.
Ans: *lock table table_name in {share | exclusive} mode*
[wait [no_of_seconds] | nowait]

48. Why retrieve is usually fast when table has cluster index?
Ans: because cluster leaf page will store the datarow along with the indexed column...so when u fetch records based on the clustered column...the entire row will be displayed from the leafpage where as in case of non clustered there is a link point from the indexed column in the leaf page to another data page...where it had to travel.

49. What is primary key?
Ans: The primary key of a relational table uniquely identifies each record in the table. It can either be a normal attribute that is guaranteed to be unique.

Example:
```
CREATE TABLE employee
(emp_id  int ,
emp_name char(50),
emp_addess varchar(255))
```

We have a table named 'employee' in our database now.

Add primary key to table 'employee' on column name 'emp_id'
```
ALTER TABLE employee
ADD CONSTRAINT pk_emp
PRIMARY KEY CLUSTERED (emp_id)
```

Insert a record in 'employee' table:
```
INSERT INTO employee (emp_id, emp_name, emp_addess)
VALUES (1, 'ABHI', 'Singapore')
```

Try to insert another record with already inserted emp_id .i.e emp_id = 1
```
INSERT INTO employee (emp_id, emp_name, emp_addess)
VALUES (1, 'ABHI', 'Singapore')
```

It will give below error because 'employee' table has primary key on column 'emp_id' and primary key will not allow to insert emp_id = 1 again as its duplicate.
Error: Number (2601) Severity (14) State (1) Server (SERVER_NAME) Attempt to insert duplicate key row in object 'employee' with unique index 'pk_emp'Attempt to insert duplicate key row in object 'employee' with unique index 'pk_emp'

We can not insert 'NULL' value to a primary key column
INSERT INTO employee (emp_id, emp_name, emp_addess)
VALUES (NULL, 'ABHI', 'Singapore')
Error: Number (233) Severity (16) State (1) Server (SERVER_NAME) The column emp_id in table employee does not allowed

50. What are the types of indexes in Sybase?
Ans:
Clustered indexes, where the table data is physically stored in the order of the keys on the index:
Nonclustered indexes, where the storage order of data in the table is not
You can create only one clustered index on a table because there is only one possible physical ordering of the data rows. You can create up to 249 nonclustered indexes per table.

51. What is the use of 'return' keyword in stored procedure?
Ans: Return values indicate a return code from the stored procedure. Mention return @xx where xx is a variable and declared as an out variable.

52. Can we create a temp table in trigger script?
Ans: No

53. What are the types of temporary tables in Sybase?
Ans: Session level shared temp table and Global shared table.

54. Please explain global temporary table.
Ans: A created global temporary table is a table that exists in the database like a base table and remains in the database until it is explicitly removed by a DROP TABLE statement

55. Can we create a primary key with non clustered index?
Ans: Yes

56. What are Magic Tables?
Ans: the magic tables are deleted and inserted.

57. What is global variable?
Ans: Global variables are system-supplied, predefined variables. They are distinguished from local variables by the two @ signs preceding their names--for example, *@@error*. The two @ signs are considered part of the identifier used to define the global variable.

58. What is *@@error* global variable?
Ans: The *@@error* global variable is commonly used to check the error status of the most recently executed batch in the current user session. *@@error* contains 0 if the last transaction succeeded; otherwise *@@error* contains the last error number generated by the system. A statement such as **if @@error != 0** followed by **return** causes an exit on error.

59. What is the use of *@@sqlstatus?*

Ans: *@@sqlstatus* contains status information resulting from the last **fetch** statement for the current user session. *@@sqlstatus* may contain the following values:

60. Explain Constraints in Sybase?

Ans: *Contraints are used to define primary keys, enforce uniqueness, and to describe foreign key relationships.*

Note: unique or primary key constraints create indexes upon creation.

- Primary Key
- Foreign key (Referential Integrity)
- Not Null Constraint
- Check Constraint
- Default Constraint
- Unique Key

61. What is SQL derived table**?**

Ans:

A derived table is defined by the evaluation of a query expression and differs from a regular table in that it is **neither** described in **system catalogs nor stored on disk**. In Adaptive Server, a derived table may be a **SQL derived table** or **an abstract plan derived table**.

A SQL derived table: defined by one or more tables through the evaluation of a query expression. A SQL derived table is used in the query expression in which it is defined and exists only for the duration of the query. See the Transact-SQL User's Guide.

Example:

select city from (select city from publishers) cities

62. What is Normalization?

Ans: Normalization is the process of efficiently organizing data in a database. There are two main goals of the normalization process: eliminating redundant data (for example, storing the same data in more than one table) and ensuring data dependencies make sense (only storing related data in a table). Both of these are worthy goals as they reduce the amount of space a database consumes and ensure that data is logically stored.

When a table is normalized, the non-key columns depend on the key used. From a relational model point of view, it is standard to have tables that are in Third Normal Form. Normalized physical design provides the greatest ease of maintenance, and databases in this form are clearly understood by developers.

63. What are the advantages of Normalization?

Ans:

Normalization produces smaller tables with smaller rows:

- More rows per page (less logical I/O)
- More rows per I/O (more efficient)
- More rows fit in cache (less physical I/O)
- Searching, sorting, and creating indexes is faster, since tables are narrower, and more rows fit on a data page.
- You usually have more tables. You can have more clustered indexes (one per table), so you get more flexibility in tuning queries.
- Index searching is often faster, since indexes tend to be narrower and shorter.
- More tables allow better use of segments to control physical placement of data.

- You usually have fewer indexes per table, so data modification commands are faster.
- Fewer null values and less redundant data, making your database more compact.
- Triggers execute more quickly if you are not maintaining redundant data.
- Data modification anomalies are reduced.
- Normalization is conceptually cleaner and easier to maintain and change as your needs change.
- While fully normalized databases require more joins, joins are generally very fast if indexes are available on the join columns.

Adaptive Server is optimized to keep higher levels of the index in cache, so each join performs onlyone or two physical I/Os for each matching row. The cost of finding rows already in the data cache is extremely low.

64. What are the types of temporary tables in Sybase ASE and where do we create it?

Ans: Temporary tables are created in tempdb database (one of the default system database).

There are two types of Temporary tables:

- Tables that can be shared among Adaptive Server sessions
- Tables that are accessible only by the current Adaptive Server session or procedure

65. What are types of indexes Sybase ASE 15.o supports? Explain each type.

Ans: Adaptive Server supports local and global indexes.

- A *local index* – spans data in exactly one data partition. For semantically partitioned tables, a local index has partitions that are equipartitioned with their base table; that is, the table and index share the same partitioning key and partitioning type. For all partitioned tables with local indexes, each local index partition has one and only one corresponding data partition.

  Each local index spans just one data partition. You can create local indexes on range-, hash-, list-, and round-robin–partitioned tables. Local indexes allow multiple threads to scan each data partition in parallel, which can greatly improve performance.
- A *global index* – spans all data partitions in a table. Sybase supports only unpartitioned global indexes. All unpartitioned indexes on unpartitioned tables are global.

66. What are the advantages of Local index?

Ans:

- Local indexes can increase concurrency through multiple index access points, which reduces root-page contention.
- You can place local nonclustered index subtrees (index partitions) on separate segments to increase I/O parallelism.
- You can run reorg rebuild on a per-partition basis, reorganizing the local index sub-tree while minimizing the impact on other operations.

67. What is deadlock?

Ans:

A **deadlock** occurs when two user processes each have a lock on a separate data page, index page, or table and each wants to acquire a lock on same page or table locked by the other process.

When this happens, the first process is waiting for the second release the lock, but the second process will not release it until the lock on the first process's object is released.
When tasks deadlock in Adaptive Server, a deadlock detection mechanism rolls back one of the transactions, and sends messages to the user and to the Adaptive Server error log. It is possible to induce application-side deadlock situations in which a client opens multiple connections, and these client connections wait for locks held by the other connection of the same application.
Example.

T19	Event Sequence	T20
begin transaction	T19 and T20 start.	begin transaction
update savings set balance = balance - 250 where acct_number = 25	T19 gets exclusive lock on savings while T20 gets exclusive lock on checking.	
		Update checking set balance = balance - 75 where acct_number = 45
update checking set balance = balance + 250 where acct_number = 45	T19 waits for T20 to release its lock while T20 waits for T19 to release its lock; deadlock occurs.	
commit transaction		Update savings set balance = balance + 75 where acct_number = 25