1. Using which language can a user request information from a database? a) Query b) Relational c) Structural d) Compiler

Ans: Query

1. Which one of the following is a procedural language? a) Domain relational calculus b) Tuple relational calculus c) Relational algebra d) Query language

Ans: Relational Algebra

1. The\_\_\_\_\_ operation allows the combining of two relations by merging pairs of tuples, one from each relation, into a single tuple. a) Select b) Join c) Union d) Intersection

Ans: Join

1. The \_\_\_\_\_\_\_operation performs a set union of two “similarly structured” tables a) Union b) Join c) Product d) Intersect

Ans: Union

1. The most commonly used operation in relational algebra for projecting a set of tuple from a relation is a) Join b) Projection c) Select d) Union

Ans: Select

1. The most commonly used operation in relational algebra for projecting a set of tuple from a relation is a) Join b) Projection c) Select d) Union

Ans: Select

1. A \_\_\_\_\_\_\_\_ is a pictorial depiction of the schema of a database that shows the relations in the database, their attributes, and primary keys and foreign keys. a) Schema diagram b) Relational algebra c) Database diagram d) Schema flow

Ans: Schema Diagram

1. The \_\_\_\_\_\_\_\_\_ provides a set of operations that take one or more relations as input and return a relation as an output. a) Schematic representation b) Relational algebra c) Scheme diagram d) Relation flow

Ans: Relational Algebra

**Q9. Define database model.**

A1. A database model is a type of data model that determines the logical structure of database and fundamentally determines in which manner data can be stored, organized and manipulated. The most popular example of a database model is the relational which uses a table-based format. Data Models are fundamental entities to introduce abstraction in a DBMS.

The very first data model could be flat data-models, where all the data used are to be kept in the same plane. Earlier data models were not so scientific; hence they were prone to introduce lots of duplication and update anomalies. Data models define how the logical structure of a database is framed. Data Models are fundamental entities to introduce abstraction in a DBMS. Data models define how data is connected to each other and how they are processed and stored inside the system.

Managing large quantities of structured and [unstructured data](https://en.wikipedia.org/wiki/Unstructured_data) is a primary function of [information systems](https://en.wikipedia.org/wiki/Information_system). Data models describe the structure, manipulation and integrity aspects of the data stored in data management systems such as relational databases. They typically do not describe unstructured data, such as [word processing](https://en.wikipedia.org/wiki/Word_processor) documents, [email messages](https://en.wikipedia.org/wiki/Email), pictures, digital audio, and video.

The ANSI/SPARC [three level architecture](https://en.wikipedia.org/wiki/Three_schema_approach). This shows that a data model can be an external model (or view), a conceptual model, or a physical model. This is not the only way to look at data models, but it is a useful way, particularly when comparing models.

**A data model instance may be one of three kinds according to**[**ANSI**](https://en.wikipedia.org/wiki/ANSI)**in 1975:**

* [**Conceptual data model**](https://en.wikipedia.org/wiki/Conceptual_data_model) **:**

Describes the semantics of a domain, being the scope of the model. For example, it may be a model of the interest area of an organization or industry. This consists of entity classes, representing kinds of things of significance in the domain, and relationship assertions about associations between pairs of entity classes.

* [**Logical data model**](https://en.wikipedia.org/wiki/Logical_data_model) **:**

Depicts the semantics, as represented by a particular data manipulation technology. This consists of descriptions of tables and columns, object-oriented classes, and XML tags, among other things.

* [**Physical data model**](https://en.wikipedia.org/wiki/Physical_data_model) **:**

Describes the physical means by which data are stored. This is concerned with partitions, CPUs, tablespaces, and the like.

**Q10. Define Normalization.**

Ans. Database normalization is the process of structuring a  in accordance with a series of so-called [normal forms](https://en.wikipedia.org/wiki/Database_normalization#Normal_forms) in order to reduce [data redundancy](https://en.wikipedia.org/wiki/Data_redundancy) and improve [data integrity](https://en.wikipedia.org/wiki/Data_integrity). It was first proposed by [Edgar F. Codd](https://en.wikipedia.org/wiki/Edgar_F._Codd) as part of his [relational model](https://en.wikipedia.org/wiki/Relational_model). Normalization entails organizing the [columns](https://en.wikipedia.org/wiki/Column_(database)) (attributes) and [tables](https://en.wikipedia.org/wiki/Relation_(database)) (relations) of a database to ensure that their [dependencies](https://en.wikipedia.org/wiki/Dependency_theory_(database_theory)) are properly enforced by database integrity constraints. It is accomplished by applying some formal rules either by a process of synthesis (creating a new database design) or decomposition (improving an existing database design).

A basic objective of the first normal form defined by Codd in 1970 was to permit data to be queried and manipulated using a "universal data sub-language" grounded in [first-order logic](https://en.wikipedia.org/wiki/First-order_logic). [SQL](https://en.wikipedia.org/wiki/SQL) is an example of such a data sub-language, albeit one that Codd regarded as seriously flawed.

The objectives of normalization beyond 1NF (first normal form) were stated as follows by Codd:

1. To free the collection of relations from undesirable insertion, update and deletion dependencies.
2. To reduce the need for restructuring the collection of relations, as new types of data are introduced, and thus increase the life span of application programs.
3. To make the relational model more informative to users.
4. To make the collection of relations neutrals to the query statistics, where these statistics are liable to change as time goes.

**Q11. Enlist the advantages of normalizing database.**

**Ans.** Normalization produces smaller tables with smaller rows. A fully normalized database allows its structure to be extended to accommodate new types of data without changing existing structure too much. As a result, applications interacting with the database are minimally affected. Normalized relations, and the relationship between one normalized relation and another, mirror real-world concepts and their interrelationships.

The benefits of normalization include: 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.

* More rows per page (less logical I/O)
* More rows per I/O (more efficient)
* More rows fit in cache (less physical I/O)

The benefits of normalization include:

* 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 only one or two physical I/Os for each matching row. The cost of finding rows already in the data cache is extremely low.

**Q12. Define Denormalization.**

**Ans.** Denormalization is a strategy that database managers use to increase the performance of a database infrastructure. It involves adding redundant data to a normalized database to reduce certain types of problems with database queries that combine data from various tables into a single table. In computing, denormalization is the process of trying to improve the read performance of a database, at the expense of losing some write performance, by adding redundant copies of data or by grouping data.

In many cases, denormalization involves creating separate tables or structures so that queries on one piece of information will not affect any other information tied to it. For instance, where more global data variables such as customer names are tied together with single purchases in a purchase history, a database administrator will want to make sure that work done on an item purchased will not incorrectly affect the entire customer account. Therefore, database handlers will separate the two pieces of information, sometimes with redundant data, so that they can be worked on separately.

Examples of denormalization techniques include:

* "Storing" the count of the "many" elements in a one-to-many relationship as an attribute of the "one" relation.
* Adding attributes to a relation from another relation with which it will be joined.
* Star schemas, which are also known as fact-dimension models and have been extended to snowflake schemas.
* Prebuilt summarization or OLAP cubes.

A denormalized data model is not the same as a data model that has not been normalized, and denormalization should only take place after a satisfactory level of normalization has taken place and that any required constraints and/or rules have been created to deal with the inherent anomalies in the design.

**Q13. Define Data Warehousing.**

**Ans.** Data warehousing is the electronic storage of a large amount of information by a business or organization. A data warehouse is designed to run query and analysis on historical data derived from transactional sources for business intelligence and data mining purposes. Data warehouses are used for analytical purposes and business reporting. Data warehouses typically store historical data by integrating copies of transaction data from disparate sources. Data warehouses can also use real-time data feeds for reports that use the most current, integrated information.

* Data warehousing is the electronic storage of a large amount of information by a business or organization.
* A data warehouse is designed to run query and analysis on historical data derived from transactional sources for business intelligence and data mining purposes.
* Data warehousing is used to provide greater insight into the performance of a company by comparing data consolidated from multiple heterogeneous sources.

A data warehouse is not necessarily the same concept as a standard database. A database is a transactional system that is set to monitor and update real-time data in order to have only the most recent data available. A data warehouse is programmed to aggregate structured data over a period of time. For example, a database might only have the most recent address of a customer, while a data warehouse might have all the addresses that the customer has lived in for the past 10 years.

The data stored in the warehouse is uploaded from the operational systems (such as marketing or sales). The data may pass through an operational data store and may require data cleansing for additional operations to ensure data quality before it is used in the DW for reporting.

Extract, transform, load (ETL) and extract, load, transform (ELT) are the two main approaches used to build a data warehouse system.

**Q14. What do you mean by Index hunting?**

**Ans.** Indexing is a way to optimize the performance of a database by minimizing the number of disk accesses required when a query is processed. It is a data structure technique which is used to quickly locate and access the data in a database. Indexes are created using a few database columns. Index hunting help in improving query performance by:

* Using query optimizer to coordinate queries with the workload.
* Observing the performance and effect of index and query distribution.

Indexes are used to quickly locate data without having to search every row in a database table every time a database table is accessed. Indexes can be created using one or more columns of a database table, providing the basis for both rapid random lookups and efficient access of ordered records.

An index is used to speed up data search and SQL query performance. The database indexes reduce the number of data pages that have to be read in order to find the specific record. ... When you insert a lot of rows into a heap table, the new records are written on data pages without a specific order.

**Q15. Enlist the disadvantages of query.**

**Ans.** Query languages are used to make queries in a database, and Microsoft Structured Query Language (SQL) is the standard. ... In a relational database, which contains records or rows of information, the SQL SELECT statement query allows the user to choose data and return it from the database to an application.

A query is a way of requesting information from the database. A database query can be either a select query or an action query. A select query is a query for retrieving data, while an action query requests additional actions to be performed on the data, like deletion, insertion, and updating.

Difficulty in Interfacing an SQL database is more complex than

adding a few lines of code. More Features Implemented in Proprietary way. Although SQL databases conform to ANSI & ISO standards, some databases go for proprietary extensions to standard SQL to ensure vendor lock-in. The disadvantages of query are:

* No indexes.
* Stored procedures are excessively compiled.
* Triggers and procedures are without SET NOCOUNT ON.
* Complicated joins making up inadequately written query.
* Cursors and temporary tables showcase a bad presentation.

**Q16. Enlist ways to efficiently code transactions.**

**Ans.** A transaction code represents a type of payment or bank transaction. You must define a transaction code for each type of payment you plan to use, such as manual checks, system checks, electronic files, and bills of exchange. You will use cash payment transaction codes in the Accounts Payable application.

Ways to efficiently code transactions:

* User input should not be allowed while transactions.
* While browsing, transactions must not be opened of data.
* Transactions must be kept as small as possible.
* Lower transaction segregation levels.
* Least information of data must be accessed while transacting**.**

A transaction is a very small unit of a program and it may contain several low-level tasks. A transaction in a database system must maintain Atomicity, Consistency, Isolation, and Durability − commonly known as ACID properties − in order to ensure accuracy, completeness, and data integrity.

**Q17. Differentiate Table Scan from Index Scan.**

**Ans**. A table scan is performed on a table which does not have an Index upon it looks at the rows in the table and an Index Scan is performed on an indexed table like the index itself. A table scan is a pretty straightforward process. When your query engine performs a table scan it starts from the physical beginning of the table and goes through every row in the table. If a row matches the criterion then it includes that into the result set.

If your table has a clustered index and you are firing a query which needs all or most of the rows i.e. query without WHERE or HAVING clause, then it uses an index scan. It works similar to the table scan, during the query optimization process, the query optimizer takes a look at the available index and chooses the best one, based on information provided in your joins and where clause, along with the statistical information database keeps. Based upon our understanding of indexes, you can now deduce the following points to summarize the difference between table scan, index scan, and index seek in a database:  
  
1) A table scan and an index scan are used when you need to retrieve all data like 90% to 100% while index seek is used when you need to retrieve data based upon some conditions like 10% of data.  
  
2) If your query doesn't have WHERE clause and your table doesn't have a clustered index then a full table scan is used, if it does have a [clustered index](http://javarevisited.blogspot.sg/2013/08/difference-between-clustered-index-and-nonclustered-index-sql-server-database.html#axzz51i34qgwV) then index scan is used.  
  
3) index scan is faster than a table scan because they look at sorted data and query optimizer know when to stop and look for another range.  
You can confirm that by actually looking at the execution plan for your query.  
In MSSQL management studio, you can see the execution plan by clicking **Ctrl + A** and then running your query.   
That's all about the difference between table scan in a database. As I told, there are only two ways to retrieve data in a database either by using a table scan or by using an index. The latter is faster in case of large tables. The choice of the index depends upon multiple things like the WHERE clause and joins in your table, the columns you are requesting, the size of tables, etc.  
  
If you feel that your query is slow, you must check the execution plan to confirm whether it’s using index seeks or index scan or table scan. Then you can optimize your query.

**Q18. Define Fragmentation.**

**Ans**. Fragmentation is done by the network layer when the maximum size of datagram is greater than maximum size of data that can be held a frame i.e., its Maximum Transmission Unit (MTU). The network layer divides the datagram received from transport layer into fragments so that data flow is not disrupted**.**

This (de)fragmentation happens in the TCP layer. In the application layer there are no more packets. If a 3000-byte packet enters an Ethernet network with a default MTU size of 1500 (for ethernet), it will be fragmented into two packets of each 1500 bytes in length.

Fragmentation refers to the condition of a disk in which files are divided into pieces scattered around the disk. Fragmentation occurs naturally when you use a disk frequently, creating, deleting, and modifying files. At some point, the operating system needs to store parts of a file in non-contiguous clusters.

Fragmentation occurs when an IP datagram traverses a network which has a maximum transmission unit (MTU) that is smaller than the size of the datagram. If a larger size datagram was to traverse an Ethernet network, it would require fragmentation to prevent it being discarded somewhere along the path.

Internal Fragmentation occurs when allotted memory blocks are of fixed size. External Fragmentation occurs when allotted memory blocks are of varying size. Internal Fragmentation occurs when a process needs more space than the size of allotted memory block or use less space.

**Q19. Differentiate Nested Loop, Hash Join and Merge Join.**

**Ans.**[**Nested Loop Join**](https://www.geeksforgeeks.org/join-algorithms-in-database/)**:**  
This is a type of physical join algorithm that is used in case of joining 2 relations. This join is an internal join technique, meaning that we cannot see the join. This is the simplest of all types of joins. This is the best-suited algorithm for small data and smaller transactions. In the case of 2 relations named R and S, the algorithm for the Nested Loop join would be as follows :

For each record x of R read in, do

Use the index on B for S

Get all the matching records (having B=x.A)

End

2. **Hash Join:**  
Hash Join is also a type of physical join algorithm that is used in case of joining two tables internally. The join being an internal join technique means that we cannot see the join. The join selection is done automatically by the query optimizer. The hash join is performed using two steps, the build, and probe. In the case of 2 relations named R and S, the algorithm for Hash join would be as follows:

Hash records of R, one by one, using A values

(Use the same M buckets and same hash function h)

Hash matching pair of records into the same bucket

End

3. **Sort merge**

It is joined by sorting the two data sets to be joined according to the join keys and then merging them together. The merge is very cheap, but the sort can be prohibitively expensive especially if the sort spills to disk. The cost of the sort can be lowered if one of the data sets can be accessed in sorted order via an index, although accessing a high proportion of blocks of a table via an index scan can also be very expensive in comparison to a full table scan.

A hash join is performed by hashing one data set into memory based on join columns and reading the other one and probing the hash table for matches. The hash join is very low cost when the hash table can be held entirely in memory, with the total cost amounting to very little more than the cost of reading the data sets. The cost rises if the hash table has to be spilled to disk in a one-pass sort, and rises considerably for a multipass sort.

(In pre-10g, outer joins from a large to a small table were problematic performance-wise, as the optimiser could not resolve the need to access the smaller table first for a hash join, but the larger table first for an outer join. Consequently, hash joins were not available in this situation).

The cost of a hash join can be reduced by partitioning both tables on the join key(s). This allows the optimiser to infer that rows from a partition in one table will only find a match in a particular partition of the other table, and for tables having n partitions the hash join is executed as n independent hash joins. This has the following effects:

1. The size of each hash table is reduced, hence reducing the maximum amount of memory required and potentially removing the need for the operation to require temporary disk space.
2. For parallel query operations the amount of inter-process messaging is vastly reduced, reducing CPU usage and improving performance, as each hash join can be performed by one pair of PQ processes.
3. For non-parallel query operations, the memory requirement is reduced by a factor of n, and the first rows are projected from the query earlier.

**Q20. What is Database partitioning?**

**Ans.** Partitioning is the database process where very large tables are divided into multiple smaller parts. By splitting a large table into smaller, individual tables, queries that access only a fraction of the data can run faster because there is less data to scan. The main of goal of partitioning is to aid in maintenance of large tables and to reduce the overall response time to read and load data for particular SQL operations.

The screen of the wizard offers to choose the option to whether to execute the script immediately by the wizard to create objects and a partition table, or to create a script and save it. A schedule for executing the script to perform the operations automatically can also be specified:

1. Select an output option window
2. The next screen of the wizard shows a review of selections made in the wizard:
3. Review summary window
4. Click the Finish button to complete the process:
5. Create Partition Wizard Progress window
6. References
7. Partitioning

Seamlessly integrate a powerful, SQL formatter into SSMS and/or Visual Studio with Apex SQL Refactor. Apex SQL Refactor is a SQL query formatter but it can also obfuscate SQL, refactor objects, safely rename objects and more – with nearly 200 customizable options.