Basics and Best Practices

RDBMS

"Without Data you're just another person with an opinion."

–W.Edward Deming

CAP vs ACID

- Brewer's CAP Theorem
  - Consistency - all nodes see the same data at the same time
  - Availability - every request receives a response about whether it succeeded or failed
    - Partition tolerance -The system continues to operate despite arbitrary partitioning due to network failures
- You can't have it all
- Though its desirable to have Consistency, High-Availability and Partition-tolerance in every system, unfortunately no system can achieve all three at the same time

CAP vs ACID

- ACID
  - Atomicity - Atomicity requires that each transaction be "all or nothing”
  - Consistency - The consistency property ensures that any transaction will bring the database from one valid state to another.
  - Isolation - The isolation property ensures that the concurrent execution of transactions results in a system state that would be obtained if transactions were executed serially.
  - Durability - The durability property ensures that once a transaction has been committed, it will remain so, even in the event of power loss, crashes, or errors.

Normalization

- Normalization
  - Normalization is the process of organizing data in a database.
  - This includes creating tables and establishing relationships to protect the data and to make the database more flexible by eliminating redundancy and inconsistent dependency.
- First Normal Form: No Repeating Groups
- Second Normal Form: Eliminate Redundant Data
- Third Normal Form: Eliminate Data Not Dependent On Key

Things to Consider

- Volume
- Variety
- Velocity

CRUD Operations

- Create
- Read
- Update
- Delete

C - CREATE

- Incorrect Data Types
  - I have seen this happen ubiquitously.Even Experienced developers and architects make this mistake frequently.
  - Example, whether you need INT or GUID.
  - Picking GUID DataType would cause lot of page splits , which in turn would cause more IOs.If you are already suffering from IO Bottleneck, these choices would pretty much kill the performance of your system as well as your queries.
    - On the other hand, if you have several hundred processes simultaneously writing to single table, GUID would help avoid hotspots.
    - Very good article on HOTSPOT Contention.
- HOTSPOT Contention issue article
- Cost of GUID Article
- GUID vs INT Data

C - CREATE

- Handling NULLS
  - In the world of RDBMS Systems, NULL occupies a special place. It is not a value, nor is it a state such as true or false, but it's an absence of any known value. NULL is used explicitly to indicate when a value is unknown, undefined, does not, or cannot exist.
- NULL Gotchas :
  - Comparing NULLs . (It depends on ANSI_NULLS settings)
  - Concatenating NULL Values. This might surprise you if you don't pay attention.
- ANSI vs UNICODE aka CHAR vs NVARCHAR
  - I have seem some system use ANSI datatypes and had hard time converting into Unicode compliant. It was like almost a year long project.
  - Ask yourself is saving space is important or having the flexibility to support UNICODE characters is important. You never know when the opportunities from the NON-ANSI languages would come. While taking the ASCII character set as its starting point, the Unicode Standard goes far beyond ASCII's limited ability to encode only the upper- and lowercase letters A through Z. It provides the capacity to encode all characters used for the written languages of the world—more than 1 million characters can be encoded. No escape sequence or control code is required to specify any character in any language.
- UNICODE – Double Byte
- ANSI – Single Byte

C - CREATE

- FILES AND FILEGROUPS:
  - We all are guilty of creating database with one file group and one data file and log file because that's the default option for SQL Server.It creates numerous problems in terms of scaling, copying and creates maintenance nightmare for scalability.
  - Create at least two file groups and make at-least 2 additional data files. That way if we want to scale, we can place the files groups in different spindle, RAID ,etc.
  - It lends itself nicely and we can take file group backups and restore it without doing the entire database backup and restore.Its common sense instead of one big bang file, lets split into multiple files and file-groups which provides nimbleness in maintenance and scaling.

R-READ RBAR

- RBAR – If you are a relational database professional, you know RBAR (Row By Agonizing Row) is performance killer.
- Relational Algebra provides optimum performance for set based operations and RDBMS are build on top of Relational Algebra concepts.It makes a world of difference when application gets all the result set and provides acknowledgment to SQL Server rather than fetching and processing one row at a time.
- Still I have seen many developers follow Row By Row Operations even-though they have the option of using set based Approach. I always wonder why?.
  - Simple Talk Article on RBAR
  - DataSet-Vs-DataReader Blog

R - READ

- SET NOCOUNT ON
  - SET NOCOUNT ON prevents the sending of DONE_IN_PROC messages to the client for each statement.This setting SET NOCOUNT to ON can provide a significant performance boost, because network traffic is greatly reduced.The setting specified by SET NOCOUNT is in effect at execute or run time and not at parse time.
- USE JOINS WHEN POSSIBLE
  - This provides SQL Server opportunity to make best choices on indexes . Use JOIN directly rather than sub-queries.
- USE UNION ALL RATHER THAN UNION IF POSSIBLE
- UNION
  - The UNION command is used to select related information from two tables. With UNION, only distinct values are selected.
- UNION ALL
  - The UNION ALL command is equal to the UNION command, except that UNION ALL selects all values.
- A UNION statement effectively does a DISTINCT on the results set. If you know that all the records returned are unique from your union, use UNION ALL instead, it gives faster results.

R-READ Accessing Objects

- Accessing Objects Inconsistently
  - Accessing objects inconsistently include frequent Deadlocks for queries that access overlapping tables and also long-term blocking for queries that access overlapping tables.
  - I have seen many developers forget this simple common sense best practice which causes nightmare deadlock scenarios which takes lot of expertise and resources to fix.
  - If you ask the question whats the need to Access Table A and then Table B , other developer accesses Table B and then Table A?. Most of the time it end up being not paying attention to this minor details.

R-READ

- DONT USE TRANSACTIONS IF NOT NEEDED
- Operate on Small Result Sets
- Limit the Number of Columns in SELECT List
- Use Highly SELECTIVE Indexes
- Avoid NonSargable Search Conditions
- Use BETWEEN rather than IN / OR Operator
- Avoid Arithmetic Operators on WHERE Clause
- Avoid FUNCTIONS on WHERE Clause
- Avoid OPTIMIZER Hints unless you absolutely know your data (JOIN Hint , INDEX Hint , FORCEPLAN Hint)

R-READ

- Columns – Force NON NULL if possible
- Force DRI if possible (Declarative Referential Integrity)
- Avoid Data Type Conversion
- Use EXISTS rather than COUNT(*) queries
- Use Indexes for Aggregate and Sort Conditions
- Use SET NOCOUNT ON
- Keep the transaction short if needed.
- Reduce LOCK Overhead
- Reduce the Logging Overhead

U-UPDATE

- Insert Only vs Update
  - If you can getaway with insert only operations and not to do any updates, that would save you ton of headache. I have experienced ton of deadlock issues on MERGE or UPSERT statements on high concurrency system.
- Again, you can have sound archiving strategy to solve space related problems.

D-DELETE

- Logical vs Physical Deletes
  - I have experienced with many databases where physical deletes are performed on important tables. Having Status column and doing the logical delete would save you ton of headache.
  - When you weigh-in pros/cons of logical vs physical deletes, logical deletes are more preferable.
  - You can devise purging or archiving strategy for inactive records. This is lot more elegant than physical deletes.

Security

- Least Privileged Accounts