

the index on that column. However, using an index hint overrides the decisions of the query optimizer, so it's generally not recommended that you use index hints either (see Chapter 19 for more information). To understand the performance impact of such a business rule, consider the following SELECT statement:

```
SELECT p.*
FROM Production.Product AS p
WHERE p.Name LIKE '%Caps';
```

In Figure 28-4, you can see that the execution plan used the index on the [Name] column, but it had to perform a scan instead of a seek. Since an index on a column with character data types (such as CHAR and VARCHAR) sorts the data values for the column on the leading-end characters, using a leading % in the LIKE condition doesn't allow a seek operation into the index. The matching rows may be distributed throughout the index rows, making the index ineffective for the search condition and thereby hurting the performance of the query.

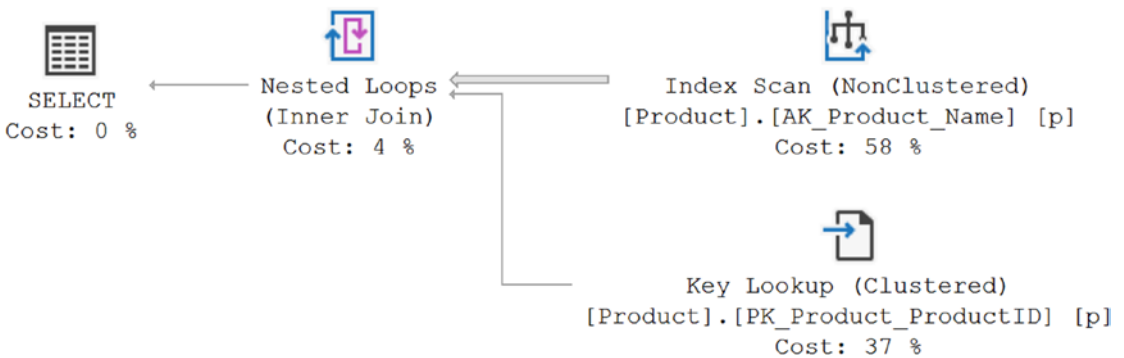


Figure 28-4. An execution plan showing a clustered index scan caused by a nonsargable LIKE clause

Avoid Arithmetic Expressions on the WHERE Clause Column

Always try to avoid using arithmetic operators and functions on columns in the WHERE and JOIN clauses. Using operators and functions on columns prevents the use of indexes on those columns. The performance impact of using arithmetic operators on WHERE clause columns is explained in detail in the “Avoid Arithmetic Operators on the WHERE Clause Column” section of Chapter 18, and the impact of using functions is explained in detail in the “Avoid Functions on the WHERE Clause Column” section of the same chapter.

To see this in action, consider the following queries:

```
SELECT soh.SalesOrderNumber
FROM Sales.SalesOrderHeader AS soh
WHERE 'S05' = LEFT(SalesOrderNumber, 3);

SELECT soh.SalesOrderNumber
FROM Sales.SalesOrderHeader AS soh
WHERE SalesOrderNumber LIKE 'S05%';
```

These queries basically implement the same logic: they check SalesOrderNumber to see whether it is equal to S05. However, the first query performs a function on the SalesOrderNumber column, while the second uses a LIKE clause to check for the same data. Figure 28-5 shows the resulting execution plans.

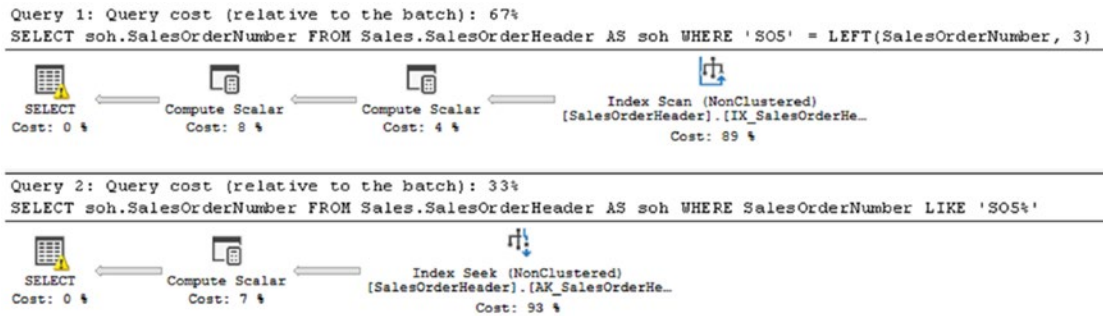


Figure 28-5. Execution plans showing a function that prevents index use

As you can see in Figure 28-5, the first query forces an Index Scan operation, while the second is able to perform a nice, clean Index Seek. These examples demonstrate clearly why you should avoid functions and operators on WHERE clause columns.

The warning you see in the plans relates to the implicit conversion occurring within the calculated columns in the SalesOrderHeader table.

Avoid Optimizer Hints

As a rule, avoid the use of optimizer hints, such as index hints and join hints, because they overrule the decision-making process of the optimizer. In most cases, the optimizer is smart enough to generate efficient execution plans, and it works best without any optimizer hint imposed on it. For a detailed understanding of the performance impact of optimizer hints, please refer to the “Avoiding Optimizer Hints” section of Chapter 19.

Stay Away from Nesting Views

A nested view exists when one view calls another view, which calls more views, and so on. This can lead to confusing code for two reasons. First, the views are masking the operations being performed. Second, the query may be simple, but the execution plan and subsequent operations by the SQL engine can be complex and expensive. This occurs because the optimizer doesn't have time to simplify the query, eliminating tables and columns it doesn't need; instead, the optimizer assumes that all tables and columns are needed. The same rule applies to nesting user-defined functions.

Ensure No Implicit Data Type Conversions

When you create variables in a query, be sure those variables are of the same data type as the columns that they will be used to compare against. Even though SQL Server can and will convert, for example, a VARCHAR to a DATE, that implicit conversion can prevent indexes from being used. You have to be just as careful in situations like table joins so that the primary key data type of one table matches the foreign key of the table being joined. You may occasionally see a warning in the execution plan to help you with this, but you can't count on this.

Minimize Logging Overhead

SQL Server maintains the old and new states of every atomic action (or transaction) in the transaction log to ensure database consistency and durability. This can place tremendous pressure on the log disk, often making the log disk a point of contention. Therefore, to improve database performance, you must try to optimize the transaction log overhead. In addition to the hardware solutions discussed later in the chapter, you should adopt the following query-design best practices:

- Choose table variables over temporary tables for small result sets, less than 20 to 50 rows, where possible. Remember, if the result set is not small, you can encounter serious issues. The performance benefit of table variables is explained in detail in the “Using Table Variables” section of Chapter 18.

- Batch a number of action queries in a single transaction. You must be careful when using this option because if too many rows are affected within a single transaction, the corresponding database objects will be locked for a long time, blocking all other users trying to access the objects.
- Reduce the amount of logging of certain operations by using the Bulk Logged recovery model. This rule applies primarily when dealing with large-scale data manipulation. You also will use minimal logging when Bulk Logged is enabled, and you use the `WRITE` clause of the `UPDATE` statement or drop or create indexes.

Adopt Best Practices for Reusing Execution Plans

The best practices for optimizing the cost of plan generation can be broadly classified into these two categories:

- Caching execution plans effectively
- Minimizing recompilation of execution plans

Caching Execution Plans Effectively

You must ensure that the execution plans for your queries are not only cached but reused often. Do so by adopting the following best practices:

- Avoid executing queries as nonparameterized, ad hoc queries. Instead, parameterize the variable parts of a query and submit the parameterized query using a stored procedure or the `sp_executesql` system stored procedure.
- If you must use lots of ad hoc queries, enable the Optimize for Ad Hoc Workload option, which will create a plan stub instead of a full plan the first time a query is called. This radically reduces the amount of procedure cache used.
- Use the same environment settings (such as `ANSI_NULLS`) in every connection that executes the same parameterized queries. This is important because the execution plan for a query is dependent on the environment settings of the connection.

- As explained earlier in the “Explicitly Define the Owner of an Object” section, explicitly qualify the owner of the objects when accessing them in your queries.

The whole idea is to ensure you have only the plans that you need in the cache and that you use those plans repeatedly rather than compiling new ones all the time. The preceding aspects of plan caching are explained in detail in Chapter 17.

Minimizing Recompilation of Execution Plans

To minimize the unnecessary generation of execution plans for queries, you must ensure that the plans in the cache are not invalidated or recompiled for reasons that are under your control. The following recommended best practices minimize the recompilation of stored procedure plans:

- Do not interleave DDL and DML statements in your stored procedures. You should put all the DDL statements at the top of the stored procedures.
- In a stored procedure, avoid using temporary tables that are created outside the stored procedure.
- Prefer table variables over temporary tables for small data sets.
- Do not change the ANSI SET options within a stored procedure.
- If you really can’t avoid a recompilation, then identify the stored procedure statement that is causing the recompilation, and execute it through the `sp_executesql` system stored procedure.

The causes of stored procedure recompilation and the recommended solutions are explained in detail in Chapter 18.

Adopt Best Practices for Database Transactions

The more effectively you design your queries for concurrency, the faster the queries will be able to complete without blocking one another. Consider the following recommendations while designing the transactions in your queries:

- Keep the scope of the transactions as short as possible. In a transaction, include only the statements that must be committed together for data consistency.

- Prevent the possibility of transactions being left open because of poor error-handling routines or application logic. Do so using the following techniques:
 - Use `SET XACTABORT ON` to ensure that a transaction is aborted or rolled back on an error condition within the transaction.
 - After executing a stored procedure or a batch of queries containing a transaction from a client code, always check for an open transaction and then roll back any open transactions using the following SQL statement:

```
IF @@TRANCOUNT > 0 ROLLBACK
```

- Use the lowest level of transaction isolation required to maintain data consistency as determined by your application requirements. The amount of isolation provided by the Read Committed isolation level, the default isolation level, is sufficient most of the time. If excessive locking is occurring, consider using the Read Committed Snapshot isolation level.

The impact of transactions on database performance is explained in detail in Chapter [20](#).

Eliminate or Reduce the Overhead of Database Cursors

Since SQL Server is designed to work with sets of data, processing multiple rows using DML statements is generally much faster than processing the rows one by one using database cursors. If you find yourself using lots of cursors, reexamine the logic to see whether there are ways you can eliminate the cursors. If you must use a database cursor, then use the database cursor with the least overhead: the `FAST_FORWARD` cursor type (generally referred to as the *fast-forward-only cursor*). You can also use the equivalent `DataReader` object in [ADO.NET](#).

The performance overhead of database cursors is explained in detail in Chapter [23](#).

Use Natively Compile Stored Procedures

In situations where you're accessing only in-memory tables, you have one additional performance enhancement open to you, which is to compile your stored procedures into a DLL that runs within the SQL Server executable. As was shown in Chapter 24, this has fairly radical performance implications. Just be sure that you call the procedures in the correct fashion passing parameters by ordinal position rather than by parameter name. Although this feels like you're breaking a best practice, it leads to better performance of the compiled procedure.

Take Advantage of Query Store for Analytical Queries

Most applications using relational databases to store their information have some degree of analytical queries. Either you have an OLTP system with a few analytical queries or you have a data warehouse or reporting system with lots of analytical queries. Take advantage of columnstore indexes in support of the queries that do lots of aggregation and analysis. A clustered columnstore is best when the majority of the queries are analytical but doesn't work as well for OLTP point look up style of query. The nonclustered columnstore index adds analysis when the majority of queries are OLTP focused, but some of them need to do analysis. In this case, it's all about picking the right tool for the job.

Summary

Performance optimization is an ongoing process. It requires continual attention to database and query characteristics that affect performance. The goal of this chapter was to provide you with a checklist of these characteristics to serve as a quick and easy reference during the development and maintenance phases of your database applications.

Index

A

Active Server Pages (ASP), [85](#)

Adaptive query processing

interleaved execution

anti-patterns, [817](#)

Clustered Index Seek and Table

Scan, [819](#)

estimated number of rows, [821](#)

execution plans, [819](#)

execution times, [821](#)

multistatement functions,
[815–817](#), [821](#)

parameter sniffing, [823](#)

properties, [820–821](#)

run a query, [818](#)

WHERE clause, [822](#)

mechanisms, [810](#)

memory grant feedback

bigTransactionHistory table,
[811–812](#)

DATABASE SCOPED

CONFIGURATION, [815](#)

DISABLE_BATCH_MODE_

MEMORY_GRANT_FEED

BACK, [815](#)

execution plan, [812](#), [814](#)

Extended Events, [811](#), [814](#)

inadequate memory, [811](#)

memory_grant_feedback_loop_
disabled, [811](#)

memory_grant_updated_by_

feedback event, [813](#), [815](#)

row mode execution, [810](#)

types, [810](#)

Ad hoc workloads

definition, [474](#)

forced parameterization, [485–488](#)

optimization, [479–481](#)

plan reusability

non reusability of existing plan, [479](#)

non reuse of existing plan, [478](#)

from procedure cache, [477](#)

sys.dm_exec_cached_plans
output, [477](#)

prepared workload, [475–476](#)

simple parameterization

autoparameterized plan, [482–485](#)

limits, [485](#)

using template, [484](#)

ALTER DATABASE command,

[339](#), [387](#), [392](#)

Atomicity, consistency, isolation, and

durability (ACID), [18](#)

Automatic index management

AdventureWorksLT, [799](#)

automatic tuning

database features, [802–803](#)

enabling, [803](#)

results, [804](#)

estimated impact view, [805–806](#)

Automatic index management (*cont.*)

- evaluation period, [808](#)
- PaaS, [799](#)
- performance recommendations and
 - tuning history, [804–805](#)
- PowerShell script, [800–802](#)
- Query Store, [802](#)
- sys.dm_db_tuning_
 - recommendations, [804](#)
- T-SQL script, [799–800](#)
- tuning history, [808–809](#)
- Validation report, [807](#)

Automatic plan correction

- enable automatic tuning
 - Azure portal, [792–795](#)
 - cache, testing, [797](#)
 - CurrentState value, [797](#)
 - desired_state value, [796](#)
 - forced plan, [798](#)
 - SQL Server 2017, [796](#)
 - sys.database_automatic_
 - tuning_options, [796](#)
 - sys.dm_db_tuning_
 - recommendations, [798](#)

Query Store, [784](#)

tuning recommendations

- AdventureWorks, [785](#)
- CPU time, [788](#)
- dbo.bigTransactionHistory
 - table, [786](#)
- execution plan, data set, [786–787](#)
- FORCE_LAST_GOOD_PLAN, [788](#)
- JSON document, [788–791](#)
- planForceDetails, [789](#)
- Query Store, [785](#)
- sys.dm_db_tuning_
 - recommendations, [788, 792](#)

B

Baseline creation

- Azure SQL Database, [102](#)
- counter log
 - data collector set, [92–93](#)
 - data logs, [93–94](#)
 - Performance Monitor, [95–96](#)
 - schedule pane, [94–95](#)
- counter number, [97](#)
- monitoring virtual and hosted
 - machines, [87](#)
- Performance Monitor graphs, [98](#)
- prefer counter logs, [97](#)
- reusable list
 - .htm file, [91](#)
 - Internet browser, [92](#)
 - Performance Monitor
 - counters, [89–92](#)
 - SQL Server, [90–91](#)
- sampling interval, [98](#)
- save counter log, [98](#)
- system behavior analysis
 - database server, [99](#)
 - log analysis, [100](#)
 - performance data, [100–101](#)
 - Performance Monitor tool, [99](#)

Blocking

- atomicity
 - dbo.ProductTest table, [635](#)
 - explicit rollback, [636–637](#)
 - INSERT statement, [634–635](#)
 - logical unit of work, [633–634](#)
 - SET XACT_ABORT ON, [635–636](#)
- consistency, [637](#)
- data access requests, [633](#)
- database connection, [632](#)
- deadlocking, [632](#)

deadly embrace, [632](#)
 durability, [639–640](#)
 information
 cause of, [680](#)
 Extended Events and blocked_
 process_report, [684–687](#)
 SQL, [681–683](#)
 isolation, [638](#)
 locking, [632](#)
 lock manager, [632](#)
 Performance Monitor
 counters, [693](#)
 reduce/avoid, recommendations,
 [692–693](#)
 resolutions
 covering index, contended data,
 [691–692](#)
 isolation level, [690](#)
 optimizing queries, [688–690](#)
 partitioning, contended data,
 [690–691](#)
 SQL Server alerts
 Blocked Process report, [694–695](#)
 and jobs, [694](#)
 SQL Server Enterprise
 Manager, [696–697](#)
 Bookmark lookups, [222, 319, 321](#)

C

Causality tracking, [126–127](#)
 CHECK constraint, [889](#)
 Checkpoint process, [60](#)
 Client cursors, [722](#)
 Client-side cursors
 characteristics, [724](#)
 cost benefits, [732](#)
 cost overhead/drawbacks, [732–733](#)

Clustered indexes, [187](#)
 creating, [213](#)
 data access, [217](#)
 data retrieval, [218–219](#)
 frequently updatable columns, [220–221](#)
 heap tables, [209](#)
 narrow, [213, 215–216](#)
 and nonclustered
 B-tree structure, [212](#)
 data page, [212](#)
 dbo.DatabaseLog, [210](#)
 execution plan, [211–212](#)
 heap table, [209](#)
 nested loop operation, [211](#)
 RID lookup operation, [211](#)
 row locator, [209–210, 212](#)
 rebuilding, [216](#)
 uniqueifier, [216–217](#)
 wide keys, [221](#)
 Clustered Index Scan, [263](#)
 Clustered IndexSeek, [263](#)
 Columnstore indexes, [769](#)
 adaptive join and attendant behavior,
 [262–263](#)
 Adaptive Threshold Rows property, [264](#)
 aggregations for GROUP BY query, [259](#)
 ALTER INDEX REORGANIZE
 command, [258](#)
 batch mode processing, [261](#)
 benefits, [257](#)
 clustered, [256, 260](#)
 Clustered Index Scan, [263](#)
 Clustered IndexSeek, [263](#)
 Columnstore Index Scan operator,
 [260–261](#)
 data types, [256](#)
 data warehousing, [256](#)

INDEX

Columnstore indexes (*cont.*)

- dbo.bigTransactionHistory, [264](#)
- deltastore, [258](#)
- dictionary, [257](#)
- make_big_adventure.sql, [257](#)
- nonclustered, [256, 259](#)
- performance enhancements, [256](#)
- reads and execution times, [260](#)
- recommendations, [266](#)
- restrictions, [256](#)
- rowgroups, [257](#)
- rowstore indexes, [257](#)
- sample query, [258](#)
- segment, [257](#)
- segment elimination, [262](#)
- status of row groups, [264](#)
- sys.dm_db_column_store_row_group_physical_stats, [264–265](#)
- tuple mover, [258](#)
- types, [259](#)

Columnstore Index Scan operator, [260–261](#)

Common table expression (CTE), [136](#)

Composite index, [200, 202](#)

Cost analysis

- client-side cursors, [732–733](#)
- dynamic cursors, [740](#)
- fast-forward-only cursor, [738](#)
- forward-only cursors, [737](#)
- keyset-driven cursors, [739](#)
- optimistic concurrency model, [735–736](#)
- read-only concurrency model, [734–735](#)
- scroll locks concurrency model, [736](#)
- server-side cursors, [733–734](#)
- static cursors, [738](#)

Cost-based optimization, [451](#)

Covering indexes, [222, 224, 231](#)

- definition, [232](#)
- HumanResources.Employee table

BusinessEntityID, [330](#)

DBCC SHOWSTATISTICS, [331–332](#)

INCLUDE columns, [328–329](#)

index storage, INCLUDE

- keyword, [329](#)

JobTitle and HireDate, [327](#)

maintenance cost, [328](#)

metrics and execution plan, [327–329](#)

NationalIDNumber, [330](#)

statistics, [332](#)

INCLUDE operator, [233, 234](#)

Index Seek operation, [234](#)

I/O and execution time, [232–233](#)

Key Lookup operator, PostalCode

- data, [233](#)

pseudoclustered index, [234](#)

recommendations, [235](#)

SELECT statement, [232](#)

CPU performance analysis

- eliminating excessive compiles/
recompiles, [76](#)

Linux, [75](#)

network analysis

- application workload, [79](#)
- Bytes Total/sec counter, [77](#)
- % Net Utilization counter, [78](#)
- Performance Monitor counters, [77](#)

optimizing application workload, [75](#)

processor analysis

- batch requests/sec, [73](#)
- context switches/sec, [72](#)
- Performance Monitor counters, [70](#)
- % Privileged Time, [72](#)
- processor queue length, [72](#)
- % Processor Time, [71](#)
- resolutions, [75–76](#)
- SQL Compilations/sec, [73](#)
- SQL Recompilations/sec, [73](#)

Query Store, [74](#)
 SQL server analysis
 batch requests/sec, [84](#)
 database concurrency, [82](#)
 Deadlocks/Sec counter, [83](#)
 dynamic management objects, [81](#)
 excessive data scans, [80](#)
 execution plan reusability, [83–84](#)
 Full Scans/sec, [80](#)
 incoming requests, [84](#)
 Lock Timeouts/sec, [82–83](#)
 Lock Wait Time (ms), [82–83](#)
 missing indexes, [80](#)
 Performance Monitor
 counters, [79–80](#)
 Total Latch Wait Time, [82](#)
 user connection, [84](#)
 Sys.dm_os_wait_stats, [74](#)
 Sys.dm_os_workers and Sys.dm_
 os_schedulers, [74](#)

Cursors

categories, [723](#)
 concurrency
 optimistic, [726–727](#)
 read-only, [725–726](#)
 scroll locks, [727](#)
 cost analysis (*see* Cost analysis)
 data manipulation, [721](#)
 default result set (*see* Default result set)
 dynamic, [730–731](#)
 events, [746](#)
 forward-only, [728](#)
 keyset-driven, [729–730](#)
 location
 client-side cursors, [724](#)
 server-side cursors, [725](#)
 Person.AddressType table, [724](#)
 positives and negatives, [740](#)

recommendations, [751–752](#)
 server, [722](#)
 static, [729](#)
 T-SQL, [722, 746–750](#)

D

Database administration
 AUTO_CLOSE, [898](#)
 AUTO_SHRINK, [899](#)
 minimum index defragmentation, [898](#)
 up-to-date statistics, [897](#)
 Database API cursor, [746](#)
 Database design
 adopting index-design, [890](#)
 configurations settings, [893](#)
 domain and referential integrity
 constraints, [887](#)
 entity-integrity constraints
 data integrity, [884](#)
 natural key, [884](#)
 UNIQUE constraint, [886](#)
 in-memory storage, [892](#)
 sp_prefix, [892](#)
 triggers, [892](#)
 use of columnstore indexes, [893](#)
 Database Engine Tuning Advisor
 advanced Tuning Options dialog
 box, [277](#)
 Apply Recommendation, [283–284](#)
 command prompt (dta.exe), [273](#)
 covering index, [283](#)
 description, [273](#)
 drop-down box, [275](#)
 limitations, [290–291](#)
 Limit Tuning Time, [276](#)
 partitioning, [276](#)
 plan cache, [288–289](#)

Database Engine Tuning Advisor (*cont.*)
 Query Store, 289–290
 query tuning general settings, 279–280
 query tuning initial recommendations, 281–282
 query tuning recommendations, 283
 reports, 278
 server and database, 274
 simple query, 279
 testing queries, 284
 tool, 271
 trace workload, 285–288
 T-SQL statements, 284
 Tuning Options tab, 275–276, 280–281
 workload, 275

Database-level lock, 647

Database performance testing
 Distributed Replay
 architecture, 828
 client configuration, 835
 execution, 836
 preprocessing, 834–835
 XML configuration file, 834

Full Recovery mode, 826

load testing, 826

playback mechanism, 826

query capture mechanism, 826

repeatable process, 827

server side trace, 829
 @DateTime, 833
 Distributed Replay, 830
 event and column, 829, 831
 profiler, 830
 SQL Server 2005–2014, 832
 standard performance test, 833
 TSQL file, 830, 832

SQL profiler, 825

SQL server 2012, 825

DATABASEPROPERTYEX function, 350

Database Transaction Unit (DTU), 102

Database workload optimization
 AdventureWorks2012 database, 843
 ALTER EVENT SESSION command, 847
 Cartesian join, 880
 costliest query identification
 baseline resource, 852
 detailed resource use, 854
 OLTP database, 851
 overall resource use, 853
 SQL workload, 852
 SSMS/query technique, 851
 worst-performing query, 851–852

CountDuration, 850

database application designs and
 database environments, 840

errors/warnings, 878

Extended Events, 847, 850

external factors analysis
 code modification, 868
 connection options, 857
 cost reduction, 867
 defragmentation (*see* Defragmentation)
 execution plan, 866
 internal behavior, 864
 lookup operation, 871–872
 processing strategy, 867
 query execution plan, 864
 statistics effectiveness, 858
 tuning, second query, 872
 wrapper procedure, 874

in-depth analysis, 849

INSERT statement, 880

Live Data explorer, 848

optimizing effect, 877

query optimization process, 841

- query types, [846–847](#)
- SELECT statement, [840, 880](#)
- server resources, [840](#)
- SLA, [878](#)
- SQL query, [850, 878–879](#)
- SQL Server performance, [842](#)
- SumDuration, [850](#)
- UPDATE statement, [840](#)
- XML field data, [849](#)
- Data Definition Language (DDL), [457](#)
- Data Manipulation Language (DML), [457](#)
- Data retrieval mechanism, [338](#)
- Data storage, [117–118](#)
- DBCC SHOW_STATISTICS command, [361, 368, 397](#)
- Deadlocks
 - access resources, physical order, [715–716](#)
 - covering index, SELECT statement, [717](#)
 - deadly embrace, [699](#)
 - error handling, [702–703](#)
 - graph, [708](#)
 - information
 - DBCC TRACEON statement, [705](#)
 - DBCC TRACESTATUS statement, [706](#)
 - execution plan, [705](#)
 - Extended Events, [704](#)
 - SQL Server Configuration Manager, [706–707](#)
 - system_health session, [703–704, 707](#)
 - trace flags, [703, 705](#)
 - lock contention
 - isolation level, [718](#)
 - locking hints, [718–719](#)
 - row versioning, [717](#)
 - lock monitor, [700](#)
 - nonclustered to clustered index, [716](#)
 - parallel operations, [700](#)
 - Purchasing.PurchaseOrderDetail table, [714](#)
 - scenario, [700](#)
 - shared lock, [700](#)
 - T-SQL statement, [708–709](#)
 - victim, [700–701](#)
 - xml:deadlock_report event, [708](#)
 - XML information, [709–714](#)
- Deadly embrace, [699](#)
- Declarative referential integrity (DRI), [600](#)
- Default result set, [738](#)
 - benefits, [742](#)
 - client-network buffer, [744](#)
 - conditions, [741](#)
 - data access layers (ADO, OLEDB, and ODBC), [741](#)
 - database requests, [744](#)
 - drawbacks, [743](#)
 - MARS, [742](#)
 - PowerShell script, [743](#)
 - sys.dm_tran_locks, [744–745](#)
 - test table, [743](#)
- Deferred object resolution, [536](#)
 - execution plan, [540](#)
 - local temporary table
 - Extended Events output, [543](#)
 - schema, [543](#)
 - stored procedure recompilation, [542](#)
 - SELECT statement, [541–542](#)
 - sql_statement_recompile event, [542](#)
 - table creation, [541](#)
- Defragmentation
 - ALTER INDEX REBUILD statement, [436](#)
 - characteristics, [443](#)
 - DROP_EXISTING clause, [432](#)
 - HumanResources.Employee table, [862](#)
 - Purchasing.PurchaseOrderHeader table, [862](#)

INDEX

Direct-attached storage (DAS), [61](#)
Disk performance analysis
 alignment, [62](#)
 Avg. Disk Sec/Read and Avg. Disk
 Sec/Write, [55](#)
 buffer manager page, [55](#)
 data files configuration, [64](#)
 disk bottleneck analysis, [51](#)
 Disk Bytes/sec counter, [54](#)
 disk counters, [52](#)
 disk transfers/sec monitors, [54](#)
 faster I/O path, [58](#)
 filegroups configuration, [63–64](#)
 I/O monitoring tools, [55](#)
 log files, [66–67](#)
 Monitoring Linux I/O, [57](#)
 new disk subsystem, [65–66](#)
 optimizing application workload, [57](#)
 PhysicalDisk and LogicalDisk
 counters, [53](#)
RAID array, [53](#)
 configurations, [58](#)
 RAID 0, [59](#)
 RAID 1, [59](#)
 RAID 1+0 (RAID 10), [61](#)
 RAID 5, [59–60](#)
 RAID 6, [60](#)
SAN system, [61](#)
solid-state drives, [62](#)
sys.dm_io_virtual_file_stats
 function, [55–56](#)
sys.dm_os_wait_stats function, [56](#)
system memory, [62](#)
table partition, [67](#)
Distributed replay administrator, [828](#)
Distributed replay client, [828](#)
Distributed replay controller, [828](#)
Domain integrity, [887](#)

DReplayClient.config file, [835](#)
Dreplay.exe command, [836](#)
DReplay.Exe.Preprocess.config file, [834](#)
DROP_EXISTING clause, [432–433](#)
Dynamic cursors
 characteristics, [730–731](#)
 cost benefits, [740](#)
 cost overhead, [740](#)
Dynamic management functions
 (DMFs), [26](#)
Dynamic management objects
 (DMOs), [26–27](#)
 sys.dm_db_xtp_table_memory_stats, [44](#)
 sys.dm_os_memory_brokers, [42](#)
 sys.dm_os_memory_clerks, [43](#)
 sys.dm_os_ring_buffers, [43–44](#)
 sys.dm_xtp_system_memory_
 consumers, [44](#)
Dynamic management views
 (DMV), [26, 463](#)

E

Entity-integrity constraints
 data integrity, [884](#)
 natural key, [884](#)
 SQL Server, [885](#)
 Stream Aggregate operation, [886](#)
 UNIQUE, [886](#)
Execution plan cache
 ad hoc workloads (*see* Ad hoc
 workloads)
 recommendations
 avoiding ad hoc queries, [506](#)
 avoiding implicit resolution,
 [508–509](#)
 parameterizing variable parts, [508](#)
 prepare/execute model, [506](#)

- query, [504](#)
 - sp_executesql coding, [505–508](#)
 - steps, [504](#)
 - stored procedure creation, [505](#)
 - reuse, [473–474](#)
 - sys.dm_exec_cached_plans, [471–472](#)
 - Execution plan generation
 - aging, [469–470](#)
 - binding
 - error statement, [455](#)
 - query processor tree, [454](#)
 - syntax-based optimization, [455–456](#)
 - warning indicator, [456](#)
 - cost-based optimization, [451](#)
 - execution context, [469](#)
 - parse tree, [454](#)
 - query compilation, [454](#)
 - query plan, [468](#)
 - relational engine, [454](#)
 - SQL Server techniques
 - query execution, [453](#)
 - resource consumption, [451](#)
 - storage engine, [454](#)
 - Extended Events sessions
 - Advanced page, [119](#)
 - automation
 - GUI, [123–124](#)
 - T-SQL, [124–126](#)
 - causality tracking, [126–127](#)
 - data storage, [117–118](#)
 - date and time, [120–121](#)
 - description, [103](#)
 - event fields
 - actions commands, [116](#)
 - configure on display, [117](#)
 - Event library, [109](#)
 - Events page, [107](#)
 - filters, [113–115](#)
 - General page, [104–105](#)
 - global fields, [111–113](#)
 - live output, wizard, [119–120](#)
 - Management Studio GUI, [104](#)
 - monitor query completion, [108](#)
 - query_hash, [121](#)
 - query performance, [109–111](#)
 - Query Store, [103](#)
 - recommendations
 - cautious with debug events, [128](#)
 - No_Event_Loss, [128](#)
 - set max file size, [127](#)
 - resource stress, [108](#)
 - RPC mechanism, [108](#)
 - system_health, [121–123](#)
 - templates, [105–106](#)
 - T-SQL batch, [108](#)
 - XE Profiler, [106](#)
 - Extent-level lock, [645](#)
 - External fragmentation, [407, 418](#)
- ## F
- Fast-forward-only cursor, [738](#)
 - Filtered indexes, [231](#)
 - ANSI settings, [246](#)
 - covering index, [243](#)
 - definition, [242](#)
 - execution plans, [245](#)
 - Index Seek, [244](#)
 - I/O and execution time, [242–243](#)
 - null values, [242, 243, 246](#)
 - performance, [245](#)
 - Sales.SalesOrderHeader
 - table, [242](#)
 - simplification, [245](#)

INDEX

Forced parameterization, [485–488](#)

Forward-only cursors

characteristics, [728](#)

cost benefits, [737](#)

drawbacks, [737](#)

FULLSCAN, [373, 389](#)

Full-text index, [267](#)

G

General Data Protection Regulation
(GDPR), [885](#)

4-Gig tuning (4GT), [49](#)

Globally unique identifiers (GUIDs), [885](#)

H

Hardware resource bottlenecks

identifying, [28](#)

memory, [30](#)

resolution, [29](#)

Hash index

bucket count, [764–766](#)

deep distribution, [765](#)

description, [764](#)

shallow distribution, [765](#)

sys.dm_db_xtp_hash_index_stats, [766](#)

unique indexes and primary keys, [765](#)

Heap or B-tree (HoBT) lock, [645](#)

I

Implicit data type conversion, [608–611](#)

INDEX hint, [595–596](#)

Index compression, [232](#)

code modification, [255](#)

CPU, [255](#)

definition, [253](#)

IX_Comp_Page_Test, [255](#)

IX_Test, [253, 255](#)

page-level, [253](#)

row-level, [253](#)

sys.dm_db_index_physical_
stats, [254](#)

Indexed views, [232](#)

AVG, [251](#)

benefit, [247](#)

computations, [250](#)

execution plan, [252](#)

logical reads, [250](#)

materialized view, [246](#)

OLTP database, [247](#)

PurchaseOrderDetail table, [250–252](#)

query execution, [248–249](#)

reporting systems, [248](#)

restrictions, [247–248](#)

SELECT statement, [246](#)

T-SQL code, [248](#)

Indexes

BIT data type column, [269](#)

B-tree structure

branch nodes, [190](#)

initial layout of 27 rows, [189](#)

ordered layout of 27 rows, [189](#)

root node, [190](#)

search process, [190](#)

single-column table, [189](#)

clustered (*see* Clustered indexes)

column data type, [204–205](#)

column order

composite index, [206](#)

execution plan, [207–208](#)

leading edge, [205, 207](#)

Seek operation, [207–208](#)

SELECT statement, [207](#)

computed columns, [268](#)

CREATE INDEX operation, [269](#)

- Database Engine Tuning Advisor
 - tool, 271
- data manipulation queries
 - DELETE statement, 191, 193
 - INSERT statement, 191
 - test table, 192
 - UPDATE statement, 191, 193–194
- definition, 185
- different column sort order, 268
- heap table, 188
- impact of, 191
- locking
 - clustered index, 678–679
 - nonclustered index, 676–678
 - resource_type, 676
 - sys.dm_tran_locks, 675
 - test table, 675
- manufacturer, 188
- MaritalStatus column
 - composite index, 200
 - DBCC SHOW_STATISTICS, 201
 - execution plan, 201–202
 - FORCESEEK, 204
 - HumanResources.Employee table, 201–203
 - Nested Loops join and Key Lookup operator, 204
 - unique values, 200
 - WHERE clause/join criterion, 201
- narrow, 197–200
- nonclustered (*see* Nonclustered indexes)
- online index creation, 270
- parallel index creation, 270
- Production.Product table, 186
- scan process, 188
- Serializable isolation level, 679–680
- StandardCost, product table, 186–187
- WHERE clause and JOIN criteria
 - columns, 195–197
- Index fragmentation
 - ALTER INDEX REBUILD statement
 - CREATE INDEX and DROP_
 - EXISTING clause, 433, 434
 - defragmentation technique, 436
 - internal and external
 - fragmentation, 435
 - PAD_INDEX setting, 436
 - sys.dm_db_index_physical_stats, 435
 - ALTER INDEX REORGANIZE
 - statement, 437–442
 - analyzing amount of, 423–424
 - automatic maintenance, database
 - analysis, 449
 - causes of, 405
 - clustered index, 419–420
 - columnstore indexes, 421–423
 - data modification and columnstore
 - indexes, 415–417
 - data modification and rowstore
 - indexes, 405
 - defragmentation and partitions, 444–445
 - disk and random I/O operation, 418
 - extents, 406, 407
 - external fragmentation, 407, 418
 - fill factor
 - Avg. Page Density (full), 448
 - avg_page_space_used_in_percent, 447
 - clustered index, 446
 - default fill factor, 446
 - INSERT and UPDATE
 - operations, 447
 - small test table, 446
 - transactional table, 445

INDEX

Index fragmentation (*cont.*)

INSERT statement

DBCC IND and DBCC PAGE, [415](#)

dbo.Test1, [414](#)

page split, [413–414](#)

sys.dm_db_index_physical_stats
output, [413](#)

internal fragmentation, [407, 418](#)

leaf pages, [405](#)

resolutions, [430](#)

DROP_EXISTING clause, [432–433](#)

dropping and re-creation, [431](#)

SELECT statements, [420–421](#)

small table analyzing, [428–430](#)

sys.dm_db_index_physical_stats

clustered index, [424](#)

detailed scan, [425, 427](#)

mixed extents, [424](#)

output, [425](#)

uniform extent, [424](#)

UPDATE statement

clustered index, [408](#)

DBCC IND output, [411–412](#)

page_count column, [410](#)

page split, [411](#)

PageType, [412](#)

SELECT statement, [409](#)

sys.dm_db_index_physical_
stats, [410](#)

Index intersections, [231](#)

covering index, [237](#)

hash join, [237](#)

key lookup, [237](#)

nonclustered index, [238](#)

OrderDate column, [235–237](#)

SalesPersonID, [235](#)

statistics I/O and time, [237](#)

Index joins, [238–241](#)

Index types

full-text, [267](#)

spatial, [267](#)

storage mechanisms, [266](#)

XML, [268](#)

In-memory OLTP tables

columnstore indexes, [769](#)

correct workload, [774](#)

data, [754](#)

database, [756–757](#)

durability, [756](#)

features, [753](#)

hash index, [764–766](#)

limitations, [757](#)

Memory Optimization Advisor (*see*

Memory Optimization Advisor)

memory-optimized technologies, [753](#)

Native Compilation Advisor, [779–781](#)

natively compiled stored procedures

dbo.CountryRegion table, [772](#)

estimated plan, [773](#)

execution time, [772](#)

parameters, [772](#)

query definition, [772](#)

SELECT operator properties, [773](#)

syntax, [771](#)

nonclustered indexes, [767–769](#)

performance baseline, [774](#)

Person.Address table

coding, [757](#)

execution plan, [762](#)

IDENTITY value, [758](#)

load data, [759, 761–762](#)

query metrics, [763](#)

query results, [760](#)

run a query, [759](#)

unsupported data types, [758](#)

spinning platters, [754](#)

- statistics maintenance, [770–771](#)
- system requirements, [755–756](#)
- transactions, [754](#)
- T-SQL code, [755](#)
- Internal fragmentation, [407, 418](#)
- Internet Information Services (IIS), [85](#)
- Internet Small Computing System
 - Interface (iSCSI), [61](#)
- Isolation levels
 - Read Committed, [661–663](#)
 - Read Uncommitted, [660–661](#)
 - repeatable read, [663, 665–667](#)
 - Serializable, [667](#)
 - Snapshot, [674](#)

J

- JOIN hint
 - execution plan, [592–593](#)
 - LOOP, [593–594](#)
 - SELECT statement, [591](#)
 - SQL Server 2017, [590–591](#)
 - types, [591](#)

K

- KEEPFIXED PLAN option, [554](#)
- Key-level lock, [643–645](#)
- Keyset-driven cursors
 - characteristics, [729–730](#)
 - cost benefits, [739](#)
 - cost overhead, [739](#)

L

- LIKE search condition, [578–579](#)
- Lock compatibility, [659](#)
- Lock escalation, [647–648](#)
- Lock granularity

- database, [647](#)
- extent, [645](#)
- HoBT, [645](#)
- KEY, [643–644](#)
- PAG, [645](#)
- resource levels, [641](#)
- RID, [642–643](#)
- TAB, [646](#)

- Lock manager, [632](#)

- Lock modes

- BU, [658](#)
- exclusive (X), [656](#)
- IS, IX, and SIX, [657–658](#)
- Key-Range, [659](#)
- resources, [648](#)
- Sch-M and Sch-S, [658](#)
- Shared (S), [649](#)
- UPDATE
 - data integrity, [650](#)
 - drawback, [655](#)
 - script order, T-SQL query windows, [650–651, 653](#)
 - sys.dm_tran_locks, [653–654](#)
 - transactions, [656](#)

- Lock monitor, [700](#)

- Lookups

- bookmark, [319, 321](#)
- clustered index, [319, 326](#)
- covering index (*see* Covering indexes)
- drawbacks, [322–323](#)
- HumanResources.Employee table
 - execution plan, [324](#)
 - key lookup Properties window, [325](#)
 - NationalIDNumber, JobTitle, and HireDate, [324](#)
 - Output List property, [325–326](#)
 - views and user-defined functions, [325](#)

INDEX

Lookups (*cont.*)

- index join (PurchaseOrderHeader)
 - covering index, [334](#)
 - execution plan, [335](#)
 - Key Lookup operation, [334](#)
 - narrow indexes, [333](#)
 - OrderDate, [334](#)
 - SELECT statement, [334](#), [335](#)
 - VendorID and OrderID, [335](#)
 - WHERE clause, [334](#)
 - nonclustered index, [319](#)
 - SalesOrderDetail table, [320–321](#)
- LOOP join hint, [593–594](#)

M

- Mapping index, [222](#)
- Memory bottleneck analysis, [894](#)
- Memory bottleneck resolutions, [894](#)
- Memory Optimization Advisor
 - data migration warnings, [776–777](#)
 - InMemoryTest database, [776](#)
 - options, [777](#)
 - Person.Address table, [775](#)
 - primary key, [778](#)
 - results, [776](#)
 - running, [776](#)
 - successful migration, [779](#)
 - unsupported data types, [775](#)
- Memory performance analysis
 - DBCC MEMORYSTATUS, [41–42](#)
 - DMO, [26–27](#), [42](#)
 - Performance Monitor tool, [24–25](#)
 - resolution
 - address fragmentation, [50](#)
 - 32-bit to 64-bit processor, [48](#)
 - data compression, [49](#)
 - flowchart, [45–47](#)

- in-memory table, [48](#)
 - memory allocation, [47](#)
 - optimizing application workload, [47](#)
 - process address space, 3GB, [49](#)
 - system memory, [48](#)
- SQL Server Management
 - Available Bytes counter, [36](#)
 - buffer cache hit ratio, [38](#)
 - buffer pool, [30](#)
 - Checkpoint Pages/sec counter, [39](#)
 - configuration, [30–31](#), [33](#)
 - dynamic memory, [34](#)
 - Lazy writes/sec counter, [39](#)
 - max server memory, [32](#)
 - Memory Grants Pending counter, [39](#)
 - memory pressure analysis, [35](#)
 - min server memory, [31](#)
 - operating system and external
 - processes, [32](#)
 - Page Faults/sec, [36–37](#)
 - Page File %Usage, [37](#)
 - Page Life Expectancy, [38–39](#)
 - Pages/sec counter, [36–37](#)
 - private bytes, [32](#)
 - RECONFIGURE statement, [34](#)
 - sp_configure system, [33](#)
 - Target and Total Server Memory, [40](#)
- Microsoft Developers Network, [138](#)
- Multiple active result sets (MARS), [742](#)
- Multiple optimization phases
 - configuration cost, [460](#)
 - DMV, [463–464](#)
 - index variations, [460](#)
 - nontrivial plan, [461](#)
 - QueryPlanHash, [463](#)
 - size and complexity, [459](#)
 - T-SQL SELECT operator, [462](#)
 - WHERE clause, [460](#)

N

- Narrow indexes, [197–200](#)
- Native Compilation Advisor, [779–781](#)
- Nonclustered indexes, [188, 767–769](#)
 - vs.* clustered indexes, [224](#)
 - analytical style queries, [228](#)
 - avoid blocking, [227](#)
 - covering index, [229–230](#)
 - credit cards, [228](#)
 - data-retrieval performance, [228](#)
 - execution plan, [226–227](#)
 - INCLUDE operation, [229](#)
 - index key size, [227](#)
 - SELECT statement, [225–227, 230](#)
 - test table, [225](#)
 - covering index, [222, 224](#)
 - frequently updatable columns, [223](#)
 - lookups, [222](#)
 - maintenance, [221–222](#)
 - mapping index, [222](#)
 - row locator, [221](#)
 - UPDATE operation, [223](#)
 - wide keys, [223](#)
- Nonsargable search conditions, [574](#)
 - BETWEEN *vs.* IN/OR, [575](#)
 - !< Condition *vs.* >= Condition, [579–580](#)
 - LIKE condition, [578–579](#)
 - and sargable conditions, [574](#)
- Nonuniform memory access (NUMA), [39](#)
- NOT NULL constraint, [597](#)

O

- Old-school approach, [523](#)
- Online index creation, [270](#)
- Online transaction processing (OLTP),
 see In-memory OLTP tables

- Optimistic concurrency model, [726–727](#)
 - benefits, [735](#)
 - cost overhead, [735–736](#)
- Optimizer hints
 - INDEX hints, [595–596](#)
 - JOIN hint, [590](#)
 - execution plan, [592–593](#)
 - LOOP join hint, [593–594](#)
 - SELECT statement, [591](#)
 - SQL Server 2017, [590–591](#)
 - types, [591](#)

P

- Page-level compression, [253](#)
- Page-level lock, [645](#)
- Parallel index creation, [270](#)
- Parallel plan optimization
 - affinity setting, [465](#)
 - cost factors, [464](#)
 - cost threshold, [466](#)
 - DML action queries, [467](#)
 - MAXDOP query hint, [465](#)
 - memory requirement, [466](#)
 - number of CPUs, [465](#)
 - OLTP queries, [467](#)
 - query execution, [467](#)
- Parameter sniffing, [620](#)
 - AddressByCity, [513](#)
 - bad parameter
 - identification, [518](#)
 - I/O and execution plan, [517](#)
 - Mentor, [517–518](#)
 - mitigating behavior, [521](#)
 - old-school approach, [523](#)
 - OPTIMIZE FOR hint, [524–525](#)
 - runtime and compile time values, [525](#)
 - SELECT properties, [524](#)

INDEX

Parameter sniffing (*cont.*)

- definition, [511](#)
- local variable, [512](#)
 - execution plan, [514–515](#)
 - query, [513–514](#)
- maintenance reexamination, [528](#)
- stored procedure, [515](#)
- sys.dm_exec_query_stats output, [516](#)
- values, [515](#)

Parse tree, [454](#)

Partition elimination, [67](#)

Performance Monitor counters, [693](#)

Performance tuning process

- baseline performance, [11–13](#)
- data access layer, [13](#)
- database connection, [5](#)
- database design, [5](#)
- hardware and software factors, [3](#)
- high level database, [15](#)
- iteration process
 - costliest query, [8–9](#)
 - user activity, [7](#)
- low level database, [15](#)
- optimization, [6](#)
- performance killers
 - cursors, [21](#)
 - excessive blocking and deadlocks, [18](#)
 - excessive fragmentation, [21](#)
 - frequent recompilation, [20](#)
 - inaccurate statistics, [17](#)
 - inappropriate database design, [19](#)
 - insufficient indexing, [16](#)
 - nonreusable execution plans, [20](#)
 - non-set-based operations, [19](#)
 - parameter sniffing, [18](#)
 - query design, [17](#)
 - SQL Server, [15–16](#)
- vs.* price, [10](#)

query optimization, [1–2](#), [6](#)

root causes, [13–14](#)

SQL Server configuration, [5](#)

Physical Design Structure (PDS), [282](#)

Plan cache, [468](#)

Plan forcing, Query Store, [313–315](#)

Plan guides

- execution plan, [562](#), [566](#)
- Index Seek operation, [566](#)
- OPTIMIZE FOR query hint, [561–563](#)
- SELECT operator property, [563](#)
- sp_create_plan_guide_from_
handle, [567](#)
- T-SQL statement, [565](#)

Platform as a Service (PaaS), [799](#)

Prepared workloads, plan reusability

- prepare/execute model, [499](#)
- sp_executesql
 - additional output, [498](#)
 - parameterized plan, [496](#)
 - plan sensitivity, [497](#)
 - SELECT statement, [495–496](#)
- stored procedures
 - extended events, [489–490](#)
 - output, [490](#)
 - reuse, [491](#)

Prepare/execute model, [475](#), [499](#), [506](#)

Pseudoclustered index, [234](#)

Q

Query analysis

- missing statistics issue
 - ALTER DATABASE command, [392](#)
 - CREATE STATISTICS statement, [394](#)
 - execution plan, [395–396](#)
 - graphical plan, [394](#)
 - Index Scan operator, [394](#)

- SELECT statement, 393
- test table creation, 392
- outdated statistics issue
 - Analyze Actual Execution Plan, 399
 - database, 400
 - DBCC SHOW_STATISTICS
 - command, 397
 - estimated *vs.* actual rows value, 398, 400
 - execution plan, 398
 - FULLSCAN, 399
 - iFirstIndex, 397
 - inaccurate_cardinality_
 - estimate, 396
 - SELECT statement, 397
 - Table Scan operator, 398
- Query compilation, 454
- Query design
 - advantage of query store, 909
 - aggregate functions, MIN and MAX, 614-615
 - arithmetic expressions, 903-904
 - compile stored procedure, 909
 - COUNT(*) and EXISTS, 611-612
 - database cursors, 908
 - database object owner, 902
 - database transactions, 907
 - implicit conversion, 905
 - implicit data type conversion, 608-611
 - local variables in batch query
 - estimated number of rows, 618-619
 - execution plan, 617-618
 - index seek and Key Lookup
 - operators, 617-618
 - I/O output, 619
 - parameter sniffing, 620
 - SELECT statement, 617
 - TransactionHistory table, 618
 - WHERE clause, 616
 - naming stored procedures, 621-623
 - nesting views, 905
 - network round-trips
 - execute multiple queries, 624
 - SET NOCOUNT, 624
 - ORDER BY clause, 616
 - nonsargable search conditions, 902-903
 - optimizer hints, 904
 - reusing execution plans, 906
 - SET NOCOUNT ON command, 902
 - techniques, 607
 - transaction cost
 - atomic action, 624
 - lock overhead, 627-629
 - logging overhead, 625-627
 - transaction log, 905-906
 - UNION ALL, 613-614
 - UNION clause, 612-613- Query design recommendations
 - avoiding optimizer hints
 - (*see* Optimizer hints)
 - domain and referential integrity
 - DRI, 600
 - NOT NULL constraint, 597
 - effective indexes
 - avoid arithmetic operators, 580
 - avoid nonsargable search conditions (*see* Nonsargable search conditions)
 - avoid WHERE clause column (*see* WHERE clause columns)
 - performance, 570
 - small result sets
 - limited number of columns, 571-572
 - WHERE clause, 572-573

INDEX

- Query execution, [105–106](#)
- Query hash, [468, 499–503](#)
- Query optimization
 - multiple phases (*see* Multiple optimization phases)
 - parallel plan optimization (*see* Parallel plan optimization)
 - simplification, [459](#)
 - statistics (*see* Statistics)
 - steps, [457–458](#)
 - trivial plan match, [459](#)
- Query performance metrics
 - Extended Events (*see* Extended Events sessions)
 - methods, [128](#)
 - sys.dm_exec_query_stats DMO, [129–130](#)
- Query plan, [468, 499–503](#)
- Query processor tree, [454](#)
- Query recompilation
 - compile process, [535](#)
 - deferred object resolution (*see* Deferred object resolution)
 - execution plan, [530, 545](#)
 - Extended Events, [533–535](#)
 - implementation
 - DDL/DML statements, [551](#)
 - disabling automatic statistics update, [556](#)
 - KEEPFIXED PLAN, [554](#)
 - OPTIMIZE FOR query hint, [559, 561](#)
 - plan guides (*see* Plan guides)
 - SET options, [558](#)
 - statistics change, [553](#)
 - table variables, [556](#)
 - index IX_Test, [531](#)
 - nonbeneficial recompilation, [532](#)
 - RECOMPILE clause (*see* RECOMPILE clause)
 - schema/binding changes, [536](#)
 - SELECT statement, [530, 532](#)
 - SET options, [544](#)
 - sp_recompile, [545](#)
 - SQL Server rules, [531](#)
 - sql_statement_recompile event, [532](#)
 - statement recompilation, [535–536](#)
 - statistics changes, [536](#)
 - stored procedure, [530](#)
- Query Store
 - behavior, [294–296](#)
 - controlling, [306–308](#)
 - monitoring, query performance, [293](#)
 - plan forcing, [313–315](#)
 - query information
 - basic structure, [297–298](#)
 - parameter definition, [299](#)
 - primary file group, [298](#)
 - query_hash value, [297](#)
 - simple parameterization, [300–301](#)
 - stored procedure, execution plan, [298](#)
 - T-SQL, [299–300](#)
 - reporting
 - AdventureWorks2017 database, [309](#)
 - details of information, [312–313](#)
 - forcing and unforcing plans, [313](#)
 - performance behaviors and execution plans, [312](#)
 - Top 25 Resource Consumers report, [310–311](#)
 - T-SQL, [309](#)
 - runtime data and wait statistics, [301–305](#)
 - safety and reporting mechanism, [293](#)
 - system views, [297](#)
 - upgrades, [316–317](#)

R

- Read Committed isolation level, [661–663](#)
- Read-only concurrency model, [725–726](#)
 - cost benefits, [734](#)
 - drawback, [735](#)
- Read Uncommitted isolation level, [660–661](#)
- Recompilation threshold (RT), [536](#)
- RECOMPILE clause
 - with CREATE PROCEDURE
 - statement, [547](#)
 - EXECUTE statement, [548–549](#)
 - query hint, [549–550](#)
- Remote procedure call (RPC), [108](#)
- Referential integrity, [887](#)
- Repeatable Read isolation level, [663](#), [665–667](#)
- Row-level compression, [253](#)
- Row-level lock, [642–643](#)
- Rowstore indexes, [257](#)

S

- Sargable search conditions, [574](#)
- Scalar functions, [586–589](#)
- Scroll locks concurrency model, [727](#)
 - benefit, [736](#)
 - cost overhead, [736](#)
- Semantic search, [267](#)
- Serializable isolation level, [679–680](#)
 - bonus payment, [668](#)
 - business functionality, [668](#)
 - clustered index, [673](#)
 - HOLDLOCK locking, [670](#)
 - PayBonus transaction, [669](#)
 - phantom read, [667](#)
 - SELECT statement, [667](#)
 - side effects, [672](#)
 - sys.dm_tran_locks, [671–673](#)

- Server-side cursors
 - characteristics, [725](#)
 - cost benefits, [733](#)
 - cost overhead/disadvantages, [734](#)
- SET statement, [544–545](#)
- Simple parameterization
 - autoparameterized plan, [482–483](#), [485](#)
 - definition, [481](#)
 - limits, [485](#)
 - using template, [484](#)
- Snapshot isolation, [674](#)
- Solid-state disks (SSDs), [51](#)
- Solid-state drives (SSD), [36](#), [62](#)
- Spatial index, [267](#)
- sp_executesql technique
 - additional output, [498](#)
 - definition, [475](#)
 - and EXECUTE, [507–508](#)
 - parameterized plan, [496](#)
 - plan sensitivity, [497](#)
 - SELECT statement, [495–496](#)
- SQL query performance
 - actual *vs.* estimated execution plans, [161](#)
 - costly query
 - multiple executions, [134](#), [137](#)
 - query execution plan, [133](#)
 - query optimizer, [133](#)
 - reads fields, [133–134](#)
 - single execution, [134–135](#)
 - slow-running query, [141–142](#)
 - execution plan
 - Active Expensive Queries, [173](#)
 - Activity Monitor, [173](#)
 - actual *vs.* estimated execution
 - plans, [142–143](#)
 - Clustered Index Scan operator, [145](#)
 - compare plans, [166](#)
 - cost-effective execution plan, [142](#)

SQL query performance (*cont.*)

- dynamic management views and functions, [164](#)
- Find Node, [165](#)
- graphical execution plan, [142–143](#)
- identification, [148, 150](#)
- index effectiveness, [150](#)
- index scan/seek, [145–146](#)
- Live Query Statistics, [170](#)
- operator selection properties, [147–148](#)
- physical and logical operation, [147](#)
- query optimizer, [142](#)
- query window, [171](#)
- scenarios, [169](#)
- SET SHOWPLAN_XML
 - command, [143](#)
- Show Execution Plan XML, [144](#)
- SQL Server Management Studio
 - 2016, [172](#)
 - tooltip sheet, [146–147](#)
 - XML execution plan, [142–143](#)
- Extended Events, [132](#)
- join effectiveness
 - adaptive, [159](#)
 - hash, [154](#)
 - merge, [156](#)
 - nested loop, [157](#)
- parameters, [133](#)
- plan cache, [164](#)
- query resource cost
 - client statistics, [176](#)
 - execution time, [178](#)
 - Extended Events, [176](#)
 - QueryTimeStats and Wait Stats, [182](#)
 - STATISTICS IO, [179](#)
- query thread profiles, [174](#)

- reducing database blocking and pressure, [132](#)

SQL Server Management Studio 2016, [172](#)

SQL server optimization

- configuration settings
 - ad hoc workloads, [895](#)
 - blocked process threshold, [895](#)
 - cost threshold, [894](#)
 - database file layout, [895–896](#)
 - data compression, [896](#)
 - max degree of parallelism, [894](#)
 - memory configuration, [894](#)
- database administration
 - AUTO_CLOSE, [898](#)
 - AUTO_SHRINK, [899](#)
 - minimum index
 - defragmentation, [898](#)
 - up-to-date statistics, [897](#)
- database backup
 - compression, [901](#)
 - distribution, [900](#)
 - transaction log frequency, [899](#)
- database design (*see* Database design)
- query design (*see* Query design)
- Statement recompilation, [535–536](#)
- Static cursors
 - characteristics, [729](#)
 - cost benefits, [738](#)
 - cost overhead, [738](#)
- Statistics
 - auto create statistics, [401](#)
 - auto update statistics, [401](#)
 - backward compatibility, [400](#)
 - cardinality estimation
 - AND calculation, [370, 372](#)
 - enabling and disabling, [377](#)
 - Estimated Row Count value, [371](#)
 - Find Node menu selection, [376](#)

- FULLSCAN, 373
- graphical execution plan, 376
- query_optimizer_estimate_
 - cardinality event, 373–374
- OR calculation, 370, 372
- stats_collection_id values, 375
- data retrieval strategy, 360–361
- definition, 337
- density, 363
- DMOs, 379
- filtered index, 367
- histogram, 359–360, 362
- iFirstIndex, 361
- maintenance behavior
 - auto create statistics, 381–382, 391
 - automatic maintenance, 381
 - auto update statistics, 382, 385
 - management setting, 387
 - manual maintenance, 385
- maintenance status, 390
- multicolumn index, 364
- nonindexed column
 - ad hoc T-SQL activities, 348
 - ALTER DATABASE command, 348
 - AUTO_CREATE_STATISTICS
 - process, 352
 - cost comparison, 357
 - execution plan, 353
 - Index Seek operation, 354
 - query optimizer, 347
 - SQL Server, 348, 355
 - sys.stats table, 352
- nonindexed columns
 - AUTO_CREATE_STATISTICS
 - OFF, 355–357
 - AUTO_CREATE_STATISTICS
 - ON, 351
 - auto_stats events, 351
- DATABASEPROPERTYEX
 - function, 350
- data distribution, 348
- FROM clause, 356–357
- graphical plan, 358
- missing_column_statistics, 359
- query cost, 357
- sample tables, 350
- SELECT statement, 350
- sys.stats_columns system, 354
- Test1.Test1_C2 and Test2.Test2_C2, 352–353
- query analysis (*see* Query analysis)
- query optimization
 - auto_stats events, 345
 - Extended Events, 341
 - indexed column, 338–339
 - large data modification, 343–344
 - nonclustered index, 340
 - outdated statistics, 345
 - SELECT statement, 340, 343
 - small data modification, 341–342
 - system resources, 339
- sampling rate, 402
- small and large result set
 - queries, 362
- Storage area network (SAN), 53, 61
- Stored procedures
 - definition, 475
 - extended events, 489–490
 - first execution, 492
 - output, 490
 - performance benefits, 492–494
 - reuse, 491
 - UserOne user, 494
- Syntax-based optimization, 455–456
- system_health Extended Events session, 121–123

INDEX

T

Table-level lock, [646](#)

T-SQL cursors, [746–750](#)

U

UNIQUE constraint, [886](#)

V

Virtual machine (VM), [24](#), [87–88](#)

W, X, Y, Z

WHERE clause

columns, [572–573](#)

custom scalar UDF,

[586–589](#)

date part comparison

CONVERT function, [585](#)

DATEPART function, [584](#)

DATETIME column, [583](#)

SUBSTRING *vs.* LIKE, [582](#)