

Index



A

- acyclic graphs. *See* directed acyclic graph (DAG)
- ad-hoc paging for row numbers, 243–244
- aggregate binding, 39
- aggregate bitwise operations
 - AND operator, 363–364
 - OR operator, 362–363
 - XOR operator, 364
- aggregations. *See also* pivoting
 - aggregate bitwise AND operator, 363–364
 - aggregate bitwise operations specialized solution, 360–362
 - aggregate bitwise OR operator, 362–363
 - aggregate bitwise XOR operator, 364
 - aggregate product specialized solution, 358–360
 - aggregate product using pivoting, 346–347
 - aggregate string concatenation specialized solution, 358
 - C# code for UDA, 348
 - calculating using OVER clause, 315–319
 - CLR code in databases, 347–348
 - creating assemblies in Visual Studio 2005, 353–357
 - CUBE option. *See* CUBE option
 - cumulative, 323–328
 - custom aggregations overview, 344–345
 - custom aggregations using pivoting, 345
 - enabling CLR and querying catalog views, 357
 - grouping factor. *See* grouping factor
 - histograms. *See* histograms
 - implementing UDA, 348
 - median value, 364–367
 - OVER clause overview, 315–316
 - overview, 315
 - ROLLUP option. *See* ROLLUP option
 - running, 321–323
 - sliding, 328–329
 - specialized solutions overview, 358
 - string concatenation using pivoting, 345–346
 - testing UDA, 357–358
 - tiebreakers, 319–321
 - UDA overview, 347
 - Visual Basic .NET code for UDA, 351
 - Year-To-Date (YTD), 330–331
- algebrizer
 - aggregate binding, 39
 - grouping binding, 39–40
 - name resolution, 38–39
 - operator flattening, 38
 - overview, 37–38
 - type derivation, 39
- algorithms, join
 - forcing a strategy, 295
 - hash, 294–295
 - loop, 291–292
 - merge, 292–293
 - overview, 291
- analytical ranking functions
 - ad-hoc paging for row numbers, 243–244
 - cursor-based solution for row numbers, 234–235
 - dense rank overview, 246
 - DENSE_RANK function overview, 246
 - IDENTITY-based solution for row numbers, 235–237
 - multipage access for row numbers, 244–245
 - nonunique sort column with tiebreaker for row numbers, 229–230
 - nonunique sort column without tiebreaker for row numbers, 230–233
 - NTILE function in SQL Server 2005, 247–249

analytical ranking functions, *continued*

NTILE function overview, 247

NTILE function set-based solutions, 249–252
overview, 222–224

paging overview for row numbers, 243

partitioning, 233–234

performance comparisons for row numbers,
237–242

RANK function overview, 246

rank overview, 246

row number overview, 224, 228–229

ROW_NUMBER function determinism,
226–227

ROW_NUMBER function overview, 224–226

ROW_NUMBER function partitioning, 227
set-based solutions for rank and dense rank,
247

set-based technique for row numbers, 227

analyzing execution plans

graphical plans, 108–115

overview, 107–108

textual showplans, 116–117

XML showplans, 117–119

analyzing trace data, 90–103

analyzing waits at instance level, 71–79

ancestors

compared to subordinates function, 485

compared to subtrees, 515

creating fn_managers function, 484–485

creating management chain (CTE solution),
486

creating management chain with two levels
(CTE solution), 486–487

limiting levels (CTE solution), 487

overview, 484

testing fn_managers function, 485–486

AND operator

aggregate bitwise, 363–364

flattening with algebrizer, 38

TOP option and APPLY table operator
solutions, 405–408

APPLY table operator

CROSS keyword, 389–390

first page solution, 403–404

matching current and previous occurrences
solution 1, 398–399

matching current and previous occurrences
solution 2, 399–400

matching current and previous occurrences
solution 3, 400–401

matching current and previous occurrences
solution 4, 401–402

matching current and previous occurrences
solution overview, 397–398

median value solution, 413–415

n rows for each group solution 1, 392–394

n rows for each group solution 2, 394

n rows for each group solution 3, 395

n rows for each group solution 4, 395–396

n rows for each group solution 5, 396–397

n rows for each group solution overview,
391–392

next page solution, 404–409

AND operator, 405–408

OUTER keyword, 390

overview, 20–22, 381, 388

paging solution overview, 402–403

passing column reference parameter, 390–391

previous page solution, 409–411

random rows solution, 411–412

TOP option WITH TIES, 389

arguments

common table expression (CTE), 215

derived tables, 213

assemblies, creating in Visual Studio 2005,
353–357

assigning left and right values in nested sets

creating relationships (CTE solution), 521–522

creating script for fn_empsnestedsets function
(UDF solution), 523

illustration of model, 518–519

- materializing nested sets relationships in tables (CTE or UDF solution), 526
- overview, 518
- producing binary sort paths (CTE solution), 519–522
- testing script for `fn_empsnestedsets` function (UDF solution), 525
- assignment SELECT statements, 450–452
- assignment UPDATE statements, 452–454
- asynchronous sequence generation, 433–434
- attributes
 - compared to dimensions, 374
 - pivoting, 331–335
- audience for this book, 4
- auxiliary table of numbers
 - creating and populating, 253–256
 - overview, 252
 - returning, 256

B

- balanced trees, 124–128
- bill of materials (BOM) scenario
 - creating script for `fn_BOMTC` (UDF solution), 533–534
 - generating all paths in BOM, 534–535
 - isolating shortest paths (CTE solution), 535–537
 - overview, 464–468
 - running code for transitive closure, 531–533
 - transitive closure overview, 531
- binding, 36
- bitwise operations
 - AND operator, 363–364
 - OR operator, 362–363
 - XOR operator, 364
- block of sequence values, 431–433
- BOM scenario. *See* bill of materials (BOM) scenario

C

- Cartesian product (cross join)
 - example, complex, 4
 - example, simple, 3–4
 - improving performance of queries, 5–7
 - overview, 265
 - performing, 6–7
- clearing cache, 105
- CLR (common language runtime), 347–348
- clustered index seek + ordered partial scans
 - index access methods, 148–150
 - index optimization scale, 161–162
- clustered indexes, 124–128
- code samples, downloading, xxiv
- common language runtime (CLR), 347–348
- common table expression (CTE)
 - ancestors, creating management chain, 486
 - ancestors, creating management chain with two levels, 486–487
 - ancestors, limiting levels, 487
 - arguments, 215
 - container objects, 218–219
 - cycles, avoiding, 503–504
 - cycles, detecting, 502–503
 - cycles, isolating paths, 504–505
 - isolating shortest paths in BOM, 535–537
 - modifying data, 217–218
 - multiple, 216
 - multiple references, 216–217
 - nested sets, creating relationships, 521–522
 - nested sets, materializing relationships in tables, 526
 - nested sets, producing binary sort paths, 519–522
 - overview, 214–215, 219–222, 472
 - result column aliases, 215
 - sorting, returning all employees sorted by empname, 499–500

- common table expression (CTE), *continued*
 - sorting, returning all employees sorted by salary, 500–501
- subgraph/subtree with path enumeration, 490–491
- subordinates, creating `fn_partsexplosion` function, 476–477
- subordinates, creating `fn_subordinates2` function, 480–481
- subordinates, creating `fn_subordinates2` function with two levels, 482
- subordinates, limiting levels, 480
- subordinates, limiting levels using filters, 483
- subordinates, limiting levels using `MAXRECURSION`, 482–483
- subordinates, parts explosion, 478
- subordinates, parts explosion with aggregate parts, 479
- subordinates, testing `fn_partsexplosion` function, 478
- subordinates, testing `fn_subordinates2` function, 481
- compilation
 - aggregate binding with algebrizer, 39
 - algebrizer overview, 37–38
 - Assert operator in update plans, 59
 - batch main steps, 35
 - batch overview, 35
 - binding, 36
 - capturing showplans with SQL trace, 55–57
 - compared to execution, 35
 - cost-based query optimizer, 40–41, 42–43
 - cost strategies in update plans, 60
 - counters of optimizer event, 47
 - creating clustered index in update plans, 62–63
 - data manipulation language (DML), 37
 - dynamic management view (DMV), 44
 - execution plans overview, 35
 - extracting showplans from procedure cache, 57–58
 - formats for showplans, 47–48
 - graphical format for showplans, 51–53
 - grouping binding with algebrizer, 39–40
 - Halloween spool in update plans, 61, 63
 - maintaining indexes in update plans, 59
 - name resolution with algebrizer, 38–39
 - operator flattening with algebrizer, 38
 - optimization overview, 36, 40
 - optimization phases, 42–43
 - outer join simplifications, 41–42
 - overview, 31, 35
 - parsing, 36
 - per-row and per-index update plans, 60–61
 - performance in update plans, 60
 - procedure cache using
 - `sys.dm_exec_query_optimizer_info`, 44
 - query plans overview, 47
 - run-time information in showplans overview, 53
 - script for batch from
 - `sys.dm_exec_query_optimizer_info`, 44–47
 - `SET STATISTICS PROFILE` in showplans, 54–55
 - `SET STATISTICS XML ON|OFF` in showplans, 53–54
 - `SHOWPLAN_ALL`, 50
 - showplans overview, 47
 - `SHOWPLAN_TEXT`, 48–50
 - simplifications, 41–42
 - stored procedure overview, 35
 - text format for showplans, 48–50
 - trivial plan optimization, 41
 - type derivation with algebrizer, 39
 - update plans overview, 59
 - update plans stages, 59
 - XML format for showplans, 50–51
- connected graphs, 460
- container objects, 218–219

- correlated subqueries
 - EXISTS, 199–207
 - overview, 195
 - tiebreaker, 196–199
 - correlating waits with queues, 80–81
 - covering nonclustered index seek + ordered partial scans
 - index access methods, 150–152
 - index optimization scale, 162
 - cross join (Cartesian product)
 - example, complex, 4
 - example, simple, 3–4
 - overview, 265
 - performance, 5–7
 - performing, 6–7
 - CTE. *See* common table expression (CTE)
 - CUBE option. *See also* ROLLUP option
 - applying, 13
 - attributes vs. dimensions, 374
 - #Cube, 377–378
 - GROUPING function, 378
 - NULL placeholder described, 375
 - NULL placeholder in the empid column, 378
 - overview, 4, 374–375
 - cumulative aggregations, 323–328
 - cursor-based solutions for row numbers, 234–235
 - custom aggregations
 - aggregate bitwise AND operator, 363–364
 - aggregate bitwise OR operator, 362–363
 - aggregate bitwise XOR operator, 364
 - aggregate product using pivoting, 346–347
 - C# code for UDA, 348
 - CLR code in databases, 347–348
 - creating assemblies in Visual Studio 2005, 353–357
 - enabling CLR and querying catalog views, 357
 - implementing UDA, 348
 - median, 364–367
 - overview, 344–345
 - pivoting, 345
 - specialized solutions for aggregate bitwise operations, 360–362
 - specialized solutions for aggregate product, 358–360
 - specialized solutions for aggregate string concatenation, 358
 - specialized solutions overview, 358
 - string concatenation using pivoting, 345–346
 - testing UDA, 357–358
 - UDA overview, 347
 - Visual Basic .NET code for UDA, 351
 - custom sequences
 - asynchronous sequence generation, 433–434
 - block of sequence values, 431–433
 - overview, 429
 - single sequence value, 430
 - synchronous sequence generation overview, 429–430
 - cycles
 - avoiding (CTE solution), 503–504
 - detecting (CTE solution), 502–503
 - isolating paths (CTE solution), 504–505
 - overview, 502
- ## D
- DAG. *See* directed acyclic graph (DAG)
 - data manipulation language (DML) statements, 37
 - data modifications
 - Assert operator in update plans, 59
 - assignment SELECT statements, 450–452
 - assignment UPDATE statements, 452–454
 - asynchronous sequence generation, 433–434
 - block of sequence values, 431–433
 - common table expression (CTE), 217–218
 - cost strategies in update plans, 60

data modifications, *continued*

- creating clustered index in update plans, 62–63
- custom sequences overview, 429
- DELETE statements using joins, 438–441
- DELETE statements with OUTPUT clause, 441–443
- DELETE vs. TRUNCATE TABLE statements, 435
- deleting data overview, 435
- execution plan for update plans, 61–62
- globally unique identifier (GUID), 434–435
- Halloween spool in update plans, 61, 63
- identity columns in sequence mechanisms, 429
- INSERT EXEC statement, 417–423
- INSERT statement with OUTPUT clause, 426–428
- inserting data overview, 417
- inserting new rows, 423–426
- maintaining indexes, 59
- overview, 417
- per-row and per-index update plans, 60–61
- performance, 60, 454–457
- removing rows with duplicate data, 435–438
- SELECT INTO statement, 417–419
- SELECT statement assignments overview, 450
- sequence mechanisms overview, 428
- single sequence value, 430
- synchronous sequence generation, 429–433
- TRUNCATE TABLE vs. DELETE statements, 435
- update plans overview, 59
- update plans stages, 59
- UPDATE statement assignments overview, 450
- UPDATE statements using joins, 443–447
- UPDATE statements with OUTPUT clause, 447–449
- updating data overview, 443

Database Engine Tuning Advisor (DTA), 121

- DELETE statements. *See also* deleting data
 - compared to TRUNCATE TABLE statement, 435
 - cost strategies, 60
 - Halloween spool, 61, 63
 - joins, 438–441
 - maintaining indexes, 59
 - OUTPUT clause, 441–443
 - per-row and per-index plans, 60–61
 - performance, 60
 - read and write stages, 59
 - TOP queries, 385–388
- deleting data. *See also* DELETE statements
 - overview, 435
 - performance considerations, 454–457
 - removing rows with duplicate data, 435–438
- dense rank
 - DENSE_RANK function overview, 246
 - overview, 246
 - set-based solutions, 247
- derived tables
 - arguments, 213
 - multiple references, 214
 - nesting, 213
 - overview, 211–212
 - result column aliases, 212–213
- determinism
 - ROW_NUMBER function, 226–227
 - TOP option, 383–384
- digraph, 460
- dimensions vs. attributes, 374
- directed acyclic graph (DAG)
 - BOM scenario, 464–468
 - compared to trees, 462
 - creating script for fn_BOMTC (UDF solution), 533–534
 - generating all paths in BOM, 534–535
 - isolating shortest paths in BOM (CTE solution), 535–537

- overview, 460
- running code for transitive closure, 531–533
- topological sort, 491
- transitive closure overview, 531
- undirected graphs. *See* undirected graphs
- directed graphs, 460
- DISTINCT clause
 - applying, 15
 - overview, 4
- DML (data manipulation language) statements, 37
- DMV (dynamic management view), 44
- drilling down
 - analyzing trace data, 90–103
 - database or file level, 82–84
 - process level, 84–85
 - trace performance workload, 85–89
- DTA (Database Engine Tuning Advisor), 121
- dynamic management objects, 106
- dynamic management view (DMV), 44, 71

E

- employee organizational chart scenario, 462–464
- enumerated path
 - overview, 487–488
 - subgraph/subtree, creating `fn_subordinates3` function, 488
 - subgraph/subtree (CTE solution), 490–491
 - subgraph/subtree, hierarchical relationships, 490
 - subgraph/subtree, testing `fn_subordinates3` function, 489
- EXCEPT set operation
 - EXCEPT ALL set operation, 306–307
 - EXCEPT DISTINCT set operation, 305
 - overview, 305
- execution plans
 - analysis overview, 107–108
 - graphical showplans, 51–53, 108–115

- textual showplans, 48–50, 116–117
- XML showplans, 50–51, 117–119
- existing ranges
 - coll vs. row number, 262
 - grouping factors, 260–261
 - overview, 256–257
 - row number vs. coll, 262
 - row numbers based on coll order, 261–262
- EXISTS predicate
 - compared to IN predicate, 200–201
 - example, 199–200
 - minimum missing value, 203–206
 - NOT EXISTS predicate vs. NOT IN predicate, 201–203
 - overview, 199
 - reverse logic applied to relational division, 206–207
- extents, 456

F

- forcing a join strategy, 295
- forests, 461
- fragmentation, index, 168–169
- FROM clause
 - overview, 4
 - performing Cartesian product (cross join), 6–7

G

- gaps
 - next pairs, 259
 - overview, 256–257
 - points before, 258
 - solutions, 257
 - starting points, 258–259
- globally unique identifier (GUID), 434–435
- graphical showplans, 51–53, 108–115

graphs

- acyclic graphs overview, 460
- BOM scenario, 464–468
- connected, 460
- cycles. *See* cycles
- directed acyclic graph (DAG). *See* directed acyclic graph (DAG)
- directed graphs overview, 460
- iteration. *See* iteration
- overview, 459–460
- resource for formal definitions, 460
- road system scenario, 468–471
- transitive closure. *See* transitive closure
- trees. *See* trees
- undirected graphs overview, 460
- GROUP BY clause. *See also* grouping factor
 - CUBE option. *See* CUBE option
 - grouping, 12–13
 - overview, 4
 - ROLLUP option. *See* ROLLUP option
- grouping binding, 39–40
- grouping factor. *See also* GROUP BY clause
 - calculating, 260–261
 - creating and populating a Stocks table, 372–374
 - difference between col1 and row number, 262
 - overview, 260, 371
 - row numbers based on col1 order, 261
- GUID (globally unique identifier), 434–435

H

- Halloween spool, 61, 63
- hash joins, 294–295
- HAVING filter
 - applying, 13–14
 - overview, 4
- heaps, 123, 128–130

hierarchies

- employee organizational chart scenario, 462–464
- overview, 459–460, 461
- resource for formal definitions, 460
- hints, 119–121
- histograms
 - altering implementation of fn_histsteps function, 371
 - generating steps, 368–369
 - overview, 367–368
 - returning with ten steps, 369–370
 - returning with ten steps and empty steps, 370
 - returning with three steps, 369
 - testing, 369
- history of SQL, 2–3
- horizontal vs. vertical operations, 303

I

- IDENTITY-based solution for row numbers, 235–237
- identity columns in sequence mechanisms, 429
- IN predicate
 - compared to EXISTS predicate, 200–201
 - NOT IN vs. NOT EXISTS predicate, 201–203
- index access methods
 - clustered index seek + ordered partial scans, 148–150
 - covering nonclustered index seek + ordered partial scans, 150–152
 - index intersections, 152–153
 - indexed views, 153–155
 - nonclustered index seek + ordered partial scan + lookups, 140–144
 - ordered clustered index scans, 136–137
 - ordered covering nonclustered index scans, 137–140
 - overview, 132
 - table scans, 132–134

- unordered clustered index scans, 132–134
- unordered covering nonclustered index scans, 134–136
- unordered nonclustered index scan + lookups, 144–148
- index fragmentations, 168–169
- index intersections, 152–153
- index optimization scale
 - analysis, 162–167
 - clustered index seek + ordered partial scans, 161–162
 - covering nonclustered index seek + ordered partial scans, 162
 - nonclustered index seek + ordered partial scan + lookups, 158–161
 - overview, 155–156
 - selectivity point, 159–161
 - summary, 162–167
 - table scans (unordered clustered index scans), 156
 - unordered covering nonclustered index scans, 157
 - unordered nonclustered index scan + lookups, 158
- index partitioning, 170
- index scans
 - clustered index seek + ordered partial scans, 148–150, 161–162
 - covering nonclustered index seek + ordered partial scans, 150–152, 162
 - nonclustered index seek + ordered partial scan + lookups, 140–144, 158–161
 - ordered clustered index scans, 136–137
 - ordered covering nonclustered index scans, 137–140
 - unordered clustered, 132–134
 - unordered covering nonclustered, 157
 - unordered covering nonclustered index scans, 134–136
 - unordered nonclustered index scan + lookups, 144–148, 158
- index structures
 - balanced trees, 124–128
 - clustered indexes, 124–128
 - extents, 456
 - heaps, 123
 - nonclustered indexes on clustered tables, 130–131
 - nonclustered indexes on heaps, 128–130
 - overview, 122
 - pages, 122–123
- index tuning. *See also* query tuning
 - balanced trees, 124–128
 - clustered index seek + ordered partial scans, 148–150, 161–162
 - clustered indexes, 124–128
 - covering nonclustered index seek + ordered partial scans, 150–152, 162
 - extents, 123
 - fragmentation, 168–169
 - heaps, 123
 - index access methods overview, 132
 - index intersections, 152–153
 - index optimization scale analysis, 162–167
 - index optimization scale overview, 155–156
 - index optimization scale summary, 162–167
 - index structures overview, 122
 - indexed views, 153–155
 - nonclustered index seek + ordered partial scan + lookups, 140–144, 158–161
 - nonclustered indexes on clustered tables, 130–131
 - nonclustered indexes on heaps, 128–130
 - ordered clustered index scans, 136–137
 - ordered covering nonclustered index scans, 137–140
 - overview, 103–105, 122
 - pages, 122–123
 - partitioning, 170
 - selectivity point, 159–161
 - table scans, 132–134

index tuning, *continued*

table scans (unordered clustered index scans),
156

table structures overview, 122

unordered clustered index scans, 132–134

unordered covering nonclustered index scans,
134–136, 157

unordered nonclustered index scan + lookups,
144–148, 158

indexed views, 153–155

inner joins

example, 271–272

overview, 270

performance, 270

inner queries. *See* subqueries

input expressions for TOP queries, 385

INSERT EXEC statement, 417–423

INSERT statements

Assert operator, 59

cost strategies, 60

Halloween spool, 61, 63

maintaining indexes, 59

OUTPUT clause, 426–428

per-row and per-index plans, 60–61

performance, 60

read and write stages, 59

TOP queries, 385–388

inserting data

asynchronous sequence generation, 433–434

block of sequence values, 431–433

custom sequences overview, 429

globally unique identifier (GUID), 434–435

identity columns in sequence mechanisms, 429

INSERT EXEC statement, 417–423

INSERT statements with OUTPUT clause,
426–428

inserting new rows, 423–426

overview, 417

performance considerations, 454–457

SELECT INTO statements, 417–419

sequence mechanisms overview, 428

single sequence value, 430

synchronous sequence generation, 429–433

inserting new rows, 423–426

INTERSECT set operation

INTERSECT ALL set operation, 308–309

INTERSECT DISTINCT set operation, 308

INTO clause, 310overview, 307–308

introduction, 1

islands

coll vs. row number, 262

grouping factors, 260–261

overview, 256–257

row number vs. coll, 262

row numbers based on coll order, 261–262

iteration

advantages of, 471

ancestors, creating fn_managers function,
484–485

ancestors, creating management chain (CTE
solution), 486

ancestors, creating management chain with two
levels (CTE solution), 486–487

ancestors function vs. subordinates function,
485

ancestors, limiting levels (CTE solution), 487

ancestors overview, 484

ancestors, testing fn_managers function,
485–486

compared to materialized paths, 505

cycles, avoiding (CTE solution), 503–504

cycles, detecting (CTE solution), 502–503

cycles, isolating paths (CTE solution), 504–505

cycles overview, 502

enumerated path overview, 487–488

overview, 471

sorting, calculating integer sort values based on
sortpath order, 496

sorting, constructing binary sort paths for each
employee, 495

- sorting, creating script for usp_sortsubs procedure, 492–495
 - sorting, generating identity values for employees in each level, 495
 - sorting, limiting levels with sort based on empname, 497
 - sorting overview, 491–492
 - sorting, returning all employees sorted by empname (CTE solution), 499–500
 - sorting, returning all employees sorted by salary (CTE solution), 500–501
 - sorting, returning attributes other than employee ID, 497–498
 - sorting, returning employees sorted by salary, 498–499
 - sorting, testing specifying empname, 496–497
 - subgraph/subtree, creating fn_subordinates3 function, 488
 - subgraph/subtree, hierarchical relationships, 490
 - subgraph/subtree overview, 487
 - subgraph/subtree, path enumeration (CTE solution), 490–491
 - subgraph/subtree, testing fn_subordinates3 function, 489
 - subordinates, creating fn_partsexplosion function (CTE solution), 476–477
 - subordinates, creating fn_subordinates1 function (UDF solution), 472–474
 - subordinates, creating fn_subordinates2 function (CTE solution), 480–481
 - subordinates, creating fn_subordinates2 function with two levels (CTE solution), 482
 - subordinates function vs. ancestors function, 485
 - subordinates, getting other attributes (UDF solution), 474–475
 - subordinates, limiting levels (CTE solution), 480
 - subordinates, limiting levels using filters (CTE solution), 483
 - subordinates, limiting levels using MAXRECURSION (CTE solution), 482–483
 - subordinates overview, 472
 - subordinates, parts explosion (CTE solution), 478
 - subordinates, parts explosion with aggregate parts (CTE solution), 479
 - subordinates, subtree of given root (CTE solution), 475–476
 - subordinates, testing fn_partsexplosion function (CTE solution), 478
 - subordinates, testing fn_subordinates1 function (UDF solution), 474
 - subordinates, testing fn_subordinates2 function (CTE solution), 481
 - transitive closure. *See* transitive closure
- ## J
- joins
 - algorithms overview, 291
 - ANSI SQL, 264
 - cross. *See* cross join (Cartesian product)
 - DELETE statements, 438–441
 - forcing a strategy, 295
 - fundamental types, 264–265
 - hash, 294–295
 - inner. *See* inner joins
 - loop, 291–292
 - merge, 292–293
 - multiple, 279–285
 - new style vs. old style, 263–264
 - nonequijoins, 277–279
 - nonsupported types, 276
 - old style vs. new style, 263–264
 - outer. *See* outer joins
 - overview, 263

joins, *continued*
 self, 276–277
 semi joins, 285–287
 separating elements problem, 296–297
 separating elements solution output, 302–303
 separating elements solution step 1, 297–298
 separating elements solution step 2, 298–299
 separating elements solution step 3, 299–300
 separating elements solution step 4, 300–302
 sliding total of previous year exercise, 287–291
 UPDATE statements, 443–447

L

logic puzzles

Alternating Lamp States. *See* logic puzzles,
 Flipping Lamp Switches
 Alternating Lamp States solution, 562
 Arithmetic Maximum Calculation, 554
 Arithmetic Maximum Calculation solution, 561
 Basic Arithmetic, 557
 Basic Arithmetic solution, 565
 Cards Facing Up, 555–557
 Cards Facing Up solution, 565
 Chocolate Bar, 552
 Chocolate Bar solution, 558
 Covering a Chessboard with Domino Tiles, 554
 Covering a Chessboard with Domino Tiles
 solution, 561
 Cutting a Stick to Make a Triangle, 555
 Cutting a Stick to Make a Triangle solution, 562
 On the Dot, 553
 On the Dot solution, 559
 Find the Pattern in the Sequence, 558
 Find the Pattern in the Sequence solution, 567
 Flipping Lamp Switches, 555
 Flipping Lamp Switches solution. *See* logic
 puzzles, Alternating Lamp States solution
 Hiking a Mountain, 557
 Hiking a Mountain solution, 566

Measuring Time by Burning Ropes, 553
 Measuring Time by Burning Ropes solution,
 561
 Medication Tablets, 551–552
 Medication Tablets solution, 558
 The Missing Buck, 555
 The Missing Buck solution, 562
 Monty Hall Problem, 556
 Monty Hall Problem solution, 563–565
 overview, 551
 Piece of Cake, 556
 Piece of Cake solution, 565
 Rectangle within a Circle, 555
 Rectangle within a Circle solution, 563
 Rectangles in a Square, 553
 Rectangles in a Square solution, 559–561
 Self-Replicating Code (Quine), 557
 Self-Replicating Code (Quine) solution, 566
 To a T, 552–553
 To a T solution, 558–559

logical query processing
 adding outer rows, 10
 APPLY table operator overview, 20–22
 applying CUBE option, 13
 applying DISTINCT clause, 15
 applying HAVING filter, 13–14
 applying ON filter (join condition), 8–10
 applying ORDER BY clause, 15–17
 applying ROLLUP option, 13
 applying TOP option, 18–19
 applying WHERE filter, 11–12
 FROM clause overview, 4
 CUBE option overview, 4
 DISTINCT clause overview, 4
 ON filter overview, 4
 GROUP BY clause overview, 4
 grouping, 12–13
 HAVING filter overview, 4
 new phases in SQL Server 2005, 19
 ORDER BY clause overview, 4

- OUTER keyword overview, 4
- OVER clause overview, 27-29
- overview, 3-4
- performing Cartesian product (cross join), 6-7
- PIVOT table operator overview, 22-24
- processing SELECT list, 14-15
- ROLLUP option overview, 4
- sample query, 4-6
- SELECT list overview, 4
- set operations overview, 29-30
- steps described, 3-4
- table operators overview, 19-20
- TOP option overview, 4
- UNPIVOT table operator overview, 24-27
- WHERE filter overview, 4
- loop joins, 291-292

M

maintaining data

- adding employees who manage no one (leaves), 506-508
- moving subtrees, 508-511
- overview, 506
- removing subtrees, 511-512

matching current and previous occurrences

- TOP option and APPLY table operator solution 1, 398-399
- TOP option and APPLY table operator solution 2, 399-400
- TOP option and APPLY table operator solution 3, 400-401
- TOP option and APPLY table operator solution 4, 401-402
- TOP option and APPLY table operator solution overview, 397-398

materialized paths

- adding employees who manage no one (leaves), 506-508
- compared to iteration/recursion, 505
- creating and populating auxiliary table of numbers, 516

- creating script for fn_splitpath function, 516
- excluding root of subtrees, 513
- joining tables, 517
- limiting levels when returning subtrees with given root, 514
- maintaining data overview, 506
- moving subtrees, 508-511
- overview, 504-506
- performance, subtrees vs. ancestors, 515
- querying overview, 512
- removing subtrees, 511-512
- returning leaf nodes under given root, 514
- returning management chain of given node, 515
- returning nodes exactly n levels under given root, 514-515
- returning subtrees with given root, 513
- testing fn_splitpath function, 517
- measuring run time of queries, 106-107
- median value
 - custom aggregations solutions, 364-367
 - TOP option and APPLY table operator solutions, 413-415
- merge joins, 292-293
- methodology for query tuning
 - analyzing trace data, 90-103
 - analyzing waits at instance level, 71-79
 - correlating waits with queues, 80-81
 - determining course of action, 81
 - drilling down to database or file level, 82-84
 - drilling down to process level, 84-85
 - overview, 69-71
 - trace performance workload, 85-89
- misbehaving subqueries, 208-209
- missing ranges
 - next pairs, 259
 - overview, 256-257
 - points before, 258
 - solutions, 257
 - starting points, 258
- missing value for EXISTS predicate, 203-206

modifying data

- Assert operator in update plans, 59
- assignment SELECT statements, 450–452
- assignment UPDATE statements, 452–454
- asynchronous sequence generation, 433–434
- block of sequence values, 431–433
- common table expression (CTE), 217–218
- cost strategies in update plans, 60
- creating clustered index in update plans, 62–63
- custom sequences overview, 429
- DELETE statements using joins, 438–441
- DELETE statements with OUTPUT clause, 441–443
- DELETE vs. TRUNCATE TABLE statements, 435
- deleting data overview, 435
- globally unique identifier (GUID), 434–435
- Halloween spool in update plans, 61, 63
- identity columns in sequence mechanisms, 429
- INSERT EXEC statement, 417–423
- INSERT statement with OUTPUT clause, 426–428
- inserting data overview, 417
- inserting new rows, 423–426
- maintaining indexes, 59
- overview, 417
- per-row and per-index update plans, 60–61
- performance, 60, 454–457
- removing rows with duplicate data, 435–438
- SELECT INTO statement, 417–419
- SELECT statement assignments overview, 450
- sequence mechanisms overview, 428
- single sequence value, 430
- synchronous sequence generation, 429–433
- TRUNCATE TABLE vs. DELETE statements, 435
- update plans overview, 59
- update plans stages, 59
- UPDATE statement assignments overview, 450
- UPDATE statements using joins, 443–447

- UPDATE statements with OUTPUT clause, 447–449
- updating data overview, 443
- multipage access for row numbers, 244–245
- multiple common table expression (CTE), 216
- multiple joins
 - controlling logical join evaluation order, 282–285
 - controlling physical join evaluation order, 279–282
 - overview, 279
- multiple references
 - common table expression (CTE), 216–217
 - derived tables, 214

N

- n rows for each group
 - TOP option and APPLY table operator solution 1, 392–394
 - TOP option and APPLY table operator solution 2, 394
 - TOP option and APPLY table operator solution 3, 395
 - TOP option and APPLY table operator solution 4, 395–396
 - TOP option and APPLY table operator solution 5, 396–397
 - TOP option and APPLY table operator solution overview, 391–392
- name resolution, 38–39
- National Institute of Standards and Technology (NIST), 460
- nested derived tables, 213
- nested sets
 - assigning left and right values overview, 518
 - creating relationships (CTE solution), 521–522
 - creating script for fn_empsnestedsets function (UDF solution), 523
 - illustration of model, 518–519

- limiting levels when returning subtrees with given root, 528
- materializing relationships in tables (CTE or UDF solution), 526
- overview, 517–518
- producing binary sort paths (CTE solution), 519–522
- querying overview, 527
- returning all ancestors of given node, 530
- returning count of subordinates of given node, 529–530
- returning leaf nodes under given root, 529
- returning subtrees of given root, 527–528
- testing script for `fn_empsnestedsets` function (UDF solution), 525
- NEWID function, 434–435
- NIST (National Institute of Standards and Technology), 460
- nonclustered indexes
 - clustered tables, 130–131
 - heaps, 128–130
 - seek + ordered partial scan + lookups, 140–144, 158–161
- nonequijoins, 277–279
- nonpartitioned IDENTITY-based solution for row numbers, 236
- nonsupported join types, 276
- nonunique sort column
 - with tiebreaker, 229–230
 - without tiebreaker, 230–233
- NOT EXISTS predicate vs. NOT IN predicate, 201–203
- NTILE function
 - overview, 247
 - set-based solutions, 249–252
 - SQL Server 2005, 247–249
- NULL placeholder
 - empid column, 378

- placeholder for CUBE and ROLLUP options, 375
- Nums table. *See* auxiliary table of numbers

O

- ON filter
 - applying, 8–10
 - overview, 4
- operator flattening, 38
- optimization
 - cost-based, 40–41, 42–43
 - counters of optimizer event, 47
 - data manipulation language (DML), 37
 - dynamic management view (DMV), 44
 - outer join simplifications, 41–42
 - overview, 36, 40
 - phases, 42–43
 - procedure cache using
 - `sys.dm_exec_query_optimizer_info`, 44
 - script for batch from
 - `sys.dm_exec_query_optimizer_info`, 44–47
 - simplifications, 41–42
 - trivial plan, 41
- optimizer. *See* query optimizer
- OR operator
 - aggregate bitwise, 362–363
 - flattening with `algebrizer`, 38
 - TOP option and APPLY table operator solutions, 405–408
- ORDER BY clause
 - applying, 15–17
 - overview, 4
 - TOP option, 383–384
- ordered clustered index scans, 136–137
- ordered covering nonclustered index scans, 137–140
- organization of this book, xxiii

outer joins

example, 272–276

overview, 4, 272

simplifications, 41–42

outer rows, adding, 10

OUTPUT clause

DELETE statements, 441–443

UPDATE statements, 447–449

OVER clause

calculating aggregates, 315–319

logical query processing phases, 27–28

ORDER BY phase, 28

overview, 27, 315–316

SELECT phase, 28

overview, 1

P

pages, 122–123

paging

ad-hoc paging for row numbers, 243–244

logical transformations, 405–408

multipage access for row numbers, 244–245

AND operator, 405–408

OR operator, 405–408

overview for row numbers, 243

TOP option and APPLY table operator first page solution, 403–404

TOP option and APPLY table operator next page solution, 404–409

TOP option and APPLY table operator previous page solution, 409–411

TOP option and APPLY table operator solution overview, 402–403

parse tree, 37

parsing, 36

partitioning

IDENTITY-based solution for row numbers, 236–237

index, 170

ROW_NUMBER function, 227

set-based technique for row numbers, 233–234

path enumeration

overview, 487–488

subgraph/subtree, creating fn_subordinates3 function, 488

subgraph/subtree (CTE solution), 490–491

subgraph/subtree, hierarchical relationships, 490

subgraph/subtree, testing fn_subordinates3 function, 489

paths, materialized

adding employees who manage no one (leaves), 506–508

compared to iteration/recursion, 505

creating and populating auxiliary table of numbers, 516

creating script for fn_splitpath function, 516

excluding root of subtrees, 513

joining tables, 517

limiting levels when returning subtrees with given root, 514

maintaining data overview, 506

moving subtrees, 508–511

overview, 504–506

performance, subtrees vs. ancestors, 515

querying overview, 512

removing subtrees, 511–512

returning leaf nodes under given root, 514

returning management chain of given node, 515

returning nodes exactly n levels under given root, 514–515

returning subtrees with given root, 513

testing fn_splitpath function, 517

performance

ancestors vs. subtrees, 515

cross joins, 5–7

data modifications, 60, 454–457

DELETE statements, 60

- index tuning. *See* index tuning
- inner joins, 270
- INSERT statements, 60
- query tuning. *See* query tuning
- row number calculation techniques, 237–242
- subtrees vs. ancestors, 515
- update plans, 60
- phases, logical query processing
 - adding outer rows, 10
 - APPLY table operator overview, 20–22
 - applying CUBE option, 13
 - applying DISTINCT clause, 15
 - applying HAVING filter, 13–14
 - applying ON filter (join condition), 8–10
 - applying ORDER BY clause, 15–17
 - applying ROLLUP option, 13
 - applying TOP option, 18–19
 - applying WHERE filter, 11–12
 - FROM clause overview, 4
 - CUBE option overview, 4
 - DISTINCT clause overview, 4
 - ON filter overview, 4
 - GROUP BY overview, 4
 - grouping, 12–13
 - HAVING filter overview, 4
 - new phases in SQL Server 2005, 19
 - ORDER BY clause overview, 4
 - OUTER keyword overview, 4
 - OVER clause overview, 27–29
 - overview, 3–4
 - performing Cartesian product (cross join), 6–7
 - PIVOT table operator overview, 22–24
 - processing SELECT list, 14–15
 - ROLLUP option overview, 4
 - sample query, 4–6
 - SELECT overview, 4
 - set operations overview, 29–30
 - steps described, 3–4
 - TOP option overview, 4
 - UNPIVOT table operator overview, 24–27
 - WHERE filter overview, 4
- physical query processing
 - algebraizer. *See* algebraizer
 - compilation. *See* compilation
 - flow of data, 32–35
 - optimization. *See* optimization
 - overview, 31
 - query compilation, 63
 - query execution, 63
 - query optimizer. *See* query optimizer
 - query plans. *See* query plans
 - update plans. *See* update plans
- PIVOT, 22–24
- pivoting. *See also* aggregations
 - aggregate product, 346–347
 - aggregating data, 337–341
 - attributes, 331–335
 - compared to unpivoting, 341–343
 - custom aggregations, 345
 - histograms. *See* histograms
 - overview, 315, 331
 - relational division, 331–337
 - string concatenation, 345–346
- precedence of set operations, 309–310
- Profiler, 55, 121
- pronunciation of SQL, 2
- puzzles
 - Alternating Lamp States. *See* puzzles, Flipping Lamp Switches
 - Alternating Lamp States solution, 562
 - Arithmetic Maximum Calculation, 554
 - Arithmetic Maximum Calculation solution, 561
 - Basic Arithmetic, 557
 - Basic Arithmetic solution, 565
 - Cards Facing Up, 555–557
 - Cards Facing Up solution, 565
 - Chocolate Bar, 552
 - Chocolate Bar solution, 558

puzzles, *continued*

- Covering a Chessboard with Domino Tiles, 554
- Covering a Chessboard with Domino Tiles solution, 561
- Cutting a Stick to Make a Triangle, 555
- Cutting a Stick to Make a Triangle solution, 562
- On the Dot, 553
- On the Dot solution, 559
- Find the Pattern in the Sequence, 558
- Find the Pattern in the Sequence solution, 567
- Flipping Lamp Switches, 555
- Flipping Lamp Switches solution. *See* puzzles, Alternating Lamp States solution
- Hiking a Mountain, 557
- Hiking a Mountain solution, 566
- Measuring Time by Burning Ropes, 553
- Measuring Time by Burning Ropes solution, 561
- Medication Tablets, 551–552
- Medication Tablets solution, 558
- The Missing Buck, 555
- The Missing Buck solution, 562
- Monty Hall Problem, 556
- Monty Hall Problem solution, 563–565
- overview, 551
- Piece of Cake, 556
- Piece of Cake solution, 565
- Rectangle within a Circle, 555
- Rectangle within a Circle solution, 563
- Rectangles in a Square, 553
- Rectangles in a Square solution, 559–561
- Self-Replicating Code (Quine), 557
- Self-Replicating Code (Quine) solution, 566
- To a T, 552–553
- To a T solution, 558–559

Q

- query compilation, 63. *See also* compilation
- query execution, 63

query optimizer

- cost-based, 40–41, 42–43
 - counters of optimizer event, 47
 - dynamic management view (DMV), 44
 - outer join simplifications, 41–42
 - overview, 3, 40
 - phases, 42–43
 - procedure cache using
 - sys.dm_exec_query_optimizer_info, 44
 - script for batch from
 - sys.dm_exec_query_optimizer_info, 44–47
 - simplifications, 41–42
 - trivial plan, 41
- query plans, 31
- capturing showplans with SQL trace, 55–57
 - extracting showplans from procedure cache, 57–58
 - formats for showplans, 47–48
 - graphical format for showplans, 51–53
 - overview, 47
 - run-time information in showplans overview, 53
 - SET STATISTICS PROFILE in showplans, 54–55
 - SET STATISTICS XML ON|OFF in showplans, 53–54
 - SHOWPLAN_ALL, 50
 - showplans overview, 47
 - SHOWPLAN_TEXT, 48–50
 - summary, 63
 - text format for showplans, 48–50
 - XML format for showplans, 50–51
- query processing
- logical. *See* logical query processing
 - physical. *See* physical query processing
- query processor tree, 37
- query tuning. *See also* index tuning
- additional resources, 187–189
 - analyzing trace data, 90–103
 - analyzing waits at instance level, 71–79

- correlating waits with queues, 80–81
- determining course of action, 81
- drilling down to database or file level, 82–84
- drilling down to process level, 84–85
- exercise based on code revisions, 181–182
- exercise using cursor-based solution, 182–183
- exercise using set-based solution, 183–187
- methodology overview, 69–71
- overview, 65
- sample data, 66–69
- sample data preparation for BigSessions table, 173–177
- sample data preparation for Sessions table, 171–173
- sample data preparation overview, 170–171
- sample data using TABLESAMPLE clause, 177–180
- set-based vs. iterative or procedural approaches, 180–181
- tools. *See* tools for query tuning
- trace performance workload, 85–89

querying, materialized paths

- creating and populating auxiliary table of numbers, 516
- creating script for fn_splitpath function, 516
- excluding root of subtrees, 513
- joining tables, 517
- limiting levels when returning subtrees with given root, 514
- overview, 512
- performance, subtrees vs. ancestors, 515
- returning leaf nodes under given root, 514
- returning management chain of given node, 515
- returning nodes exactly n levels under given root, 514–515
- returning subtrees with given root, 513
- testing fn_splitpath function, 517

querying, nested sets

- limiting levels when returning subtrees with given root, 528
- overview, 527
- returning all ancestors of given node, 530
- returning count of subordinates of given node, 529–530
- returning leaf nodes under given root, 529
- returning subtrees of given root, 527–528

R

random rows with TOP option and APPLY table operator, 411–412

rank

- overview, 246
- RANK function overview, 246
- set-based solutions, 247

recursion

- advantages of iterative solutions, 471
- ancestors, creating fn_managers function, 484–485
- ancestors, creating management chain (CTE solution), 486
- ancestors function vs. subordinates function, 485
- ancestors, creating management chain with two levels (CTE solution), 486–487
- ancestors, limiting levels (CTE solution), 487
- ancestors overview, 484
- ancestors, testing fn_managers function, 485–486
- compared to materialized paths, 505
- cycles, avoiding (CTE solution), 503–504
- cycles, detecting (CTE solution), 502–503
- cycles, isolating paths (CTE solution), 504–505
- cycles overview, 502
- enumerated path overview, 487–488
- overview, 471

recursion, *continued*

- sorting, calculating integer sort values based on sortpath order, 496
- sorting, constructing binary sort paths for each employee, 495
- sorting, creating script for usp_sortsubs procedure, 492–495
- sorting, generating identity values for employees in each level, 495
- sorting, limiting levels with sort based on empname, 497
- sorting overview, 491–492
- sorting, returning all employees sorted by empname (CTE solution), 499–500
- sorting, returning all employees sorted by salary (CTE solution), 500–501
- sorting, returning attributes other than employee ID, 497–498
- sorting, returning employees sorted by salary, 498–499
- sorting, testing specifying empname, 496–497
- subgraph/subtree, creating fn_subordinates3 function, 488
- subgraph/subtree, hierarchical relationships, 490
- subgraph/subtree overview, 487
- subgraph/subtree, path enumeration (CTE solution), 490–491
- subgraph/subtree, testing fn_subordinates3 function, 489
- subordinates, creating fn_partsexplosion function (CTE solution), 476–477
- subordinates, creating fn_subordinates1 function (UDF solution), 472–474
- subordinates, creating fn_subordinates2 function (CTE solution), 480–481
- subordinates, creating fn_subordinates2 function with two levels (CTE solution), 482
- subordinates function vs. ancestors function, 485

- subordinates, getting other attributes (UDF solution), 474–475
- subordinates, limiting levels (CTE solution), 480
- subordinates, limiting levels using filters (CTE solution), 483
- subordinates, limiting levels using MAXRECURSION (CTE solution), 482–483
- subordinates overview, 472
- subordinates, parts explosion (CTE solution), 478
- subordinates, parts explosion with aggregate parts (CTE solution), 479
- subordinates, subtree of a given root (CTE solution), 475–476
- subordinates, testing fn_partsexplosion function (CTE solution), 478
- subordinates, testing fn_subordinates1 function (UDF solution), 474
- subordinates, testing fn_subordinates2 function (CTE solution), 481
- transitive closure. *See* transitive closure
- recursive common table expression. *See* common table expression (CTE)
- relational division
 - example, 192–195
 - overview, 192
 - pivoting, 331–337
 - reverse logic, 206–207
- removing rows with duplicate data, 435–438
- result column aliases
 - common table expression (CTE), 215
 - derived tables, 212–213
- road system scenario, 468–471
- ROLLUP option. *See also* CUBE option
 - applying, 13
 - example, 379–380
 - NULL placeholder described, 375
 - overview, 4, 374, 379
- rooted trees, 461

row numbers

- ad-hoc paging, 243–244
 - cursor-based solution, 234–235
 - IDENTITY-based solution, 235–237
 - multipage access, 244–245
 - nonunique sort column with tiebreaker, 229–230
 - nonunique sort column without tiebreaker, 230–233
 - overview, 224
 - paging overview, 243
 - performance comparisons for calculation techniques, 237–242
 - ROW_NUMBER function determinism, 226–227
 - ROW_NUMBER function overview, 224–226
 - ROW_NUMBER function partitioning, 227
 - set-based technique overview, 227
 - set-based technique partitioning, 233–234
 - unique sort column for set-based technique, 228–229
- row value constructors, 444
- ROW_NUMBER function. *See also* row numbers
- determinism, 226–227
 - overview, 224–226
 - partitioning, 227
- run-time information in showplans
- overview, 53
 - SET STATISTICS PROFILE, 54–55
 - SET STATISTICS XML ON|OFF, 53–54
- running aggregations
- cumulative aggregations, 323–328
 - overview, 321–323
 - sliding aggregations, 328–329
 - Year-To-Date (YTD), 330–331

S

- sample databases, installing, xxiv
- scalar subqueries
 - example, 192
 - overview, 192
- SELECT INTO statement, 417–419
- SELECT list processing, 4, 14–15
- SELECT statements, 450–452
- SELECT TOP option
 - basic example, 382
 - DELETE statements, 385–388
 - determinism, 383–384
 - input expressions, 385
 - INSERT statements, 385–388
 - modifications, 385–388
 - ORDER BY clause, 383–384
 - overview, 381–382
 - PERCENT option example, 382–383
 - WITH TIES option, 384
 - UPDATE statements, 385–388
- selectivity point, 159–161
- self-contained subqueries
 - overview, 192
 - relational division example, 192–195
 - relational division overview, 192
 - scalar subqueries example, 192
 - scalar subqueries overview, 192
- self joins, 276–277
- semi joins, 285–287
- separating elements
 - problem, 296–297
 - solution output, 302–303
 - solution step 1, 297–298
 - solution step 2, 298–299
 - solution step 3, 299–300
 - solution step 4, 300–302

sequence mechanisms

- asynchronous sequence generation, 433–434
 - block of sequence values, 431–433
 - custom sequences overview, 429
 - globally unique identifier (GUID), 434–435
 - identity columns, 429
 - overview, 428
 - single sequence value, 430
 - synchronous sequence generation, 429–433
- set-based solutions
- dense rank, 247
 - NTILE function, 249–252
 - rank, 247
- set-based technique for row numbers
- nonunique sort column with tiebreaker, 229–230
 - nonunique sort column without tiebreaker, 230–233
 - overview, 227
 - partitioning, 233–234
 - unique sort column, 228–229
- set operations
- INTO, 310
 - EXCEPT, 305
 - EXCEPT ALL, 306–307
 - EXCEPT DISTINCT, 305
 - horizontal vs. vertical, 303
 - INTERSECT, 307–308
 - INTERSECT ALL, 308–309
 - INTERSECT DISTINCT, 308
 - overview, 29–30, 263, 303–304
 - precedence, 309–310
 - UNION, 304
 - UNION ALL, 304
 - UNION DISTINCT, 304
 - unsupported logical phases, 310–313
 - vertical vs. horizontal, 303
- SET ROW COUNT option, 385
- SHOWPLAN_ALL, 50

showplans

- analyzing textual, 116–117
 - analyzing XML, 117–119
 - capturing with SQL trace, 55–57
 - extracting from procedure cache, 57–58
 - formats, 47–48
 - graphical format, 51–53, 108–115
 - overview, 47
 - run-time information overview, 53
 - SET STATISTICS PROFILE, 54–55
 - SET STATISTICS XML ON|OFF, 53–54
 - SHOWPLAN_ALL, 50
 - SHOWPLAN_TEXT, 48–50
 - text format, 48–50, 116–117
 - XML format, 50–51, 117–119
- SHOWPLAN_TEXT, 48–50
- simplifications, 41–42
- single sequence value, 430
- sliding aggregations, 328–329
- sorting
- calculating integer sort values based on sortpath order, 496
 - constructing binary sort paths for each employee, 495
 - creating script for usp_sortsubs procedure, 492–495
 - generating identity values for employees in each level, 495
 - limiting levels with sort based on empname, 497
 - overview, 491–492
 - returning all employees sorted by empname (CTE solution), 499–500
 - returning all employees sorted by salary (CTE solution), 500–501
 - returning attributes other than employee ID, 497–498
 - returning employees sorted by salary, 498–499
 - testing using empname, 496–497

- specialized solutions for custom aggregations
 - aggregate bitwise AND, 363–364
 - aggregate bitwise operations, 360–362
 - aggregate bitwise OR, 362–363
 - aggregate bitwise XOR, 364
 - aggregate product, 358–360
 - aggregate string concatenation, 358
 - median value, 364–367
 - overview, 358
- SQL Server Management Studio (SSMS), 32
- SQL Server Profiler, 55, 121
- SQL vs. T-SQL, 1
- SSMS (SQL Server Management Studio), 32
- STATISTICS IO session option, 106
- subgraph/subtree
 - creating `fn_subordinates3` function, 488
 - hierarchical relationships, 490
 - overview, 487
 - path enumeration (CTE solution), 490–491
 - testing `fn_subordinates3` function, 489
- subordinates
 - compared to `ancestors` function, 485
 - creating `fn_partsexplosion` function (CTE solution), 476–477
 - creating `fn_subordinates1` function (UDF solution), 472–474
 - creating `fn_subordinates2` function (CTE solution), 480–481
 - creating `fn_subordinates2` function with two levels (CTE solution), 482
 - getting other attributes (UDF solution), 474–475
 - limiting levels (CTE solution), 480
 - limiting levels using filters (CTE solution), 483
 - limiting levels using `MAXRECURSION` (CTE solution), 482–483
 - overview, 472
 - parts explosion (CTE solution), 478

- parts explosion with aggregate parts (CTE solution), 479
- subtree of given root (CTE solution), 475–476
- testing `fn_partsexplosion` function (CTE solution), 478
- testing `fn_subordinates1` function (UDF solution), 474
- testing `fn_subordinates2` function (CTE solution), 481
- subqueries
 - correlated subqueries EXISTS, 199–207
 - correlated subqueries overview, 195
 - correlated subqueries tiebreaker, 196–199
 - misbehaving, 208–209
 - overview, 191
 - self-contained, 192–195
 - table expressions. *See* table expressions
 - uncommon predicates, 209–211
- support for this book, xxiv–xxv
- synchronous sequence generation
 - block of sequence values, 431–433
 - overview, 429–430
 - single sequence value, 430
- `syscacheobjects`, 105
- system requirements, xxiii

T

- T-SQL vs. SQL, 1
- table expressions
 - arguments in CTE, 215
 - arguments in derived tables, 213
 - container objects, 218–219
 - CTE overview, 214–215
 - derived tables overview, 211–212
 - modifying data in CTE, 217–218
 - multiple CTE, 216
 - multiple references in CTE, 216–217
 - multiple references in derived tables, 214

table expressions, *continued*

- nesting in derived tables, 213
- overview, 211
- recursive CTE, 219–222
- result column aliases in CTE, 215
- result column aliases in derived tables, 212–213

table operators, 19–20

table scans, 132–134, 156

table structures

- balanced trees, 124–128
- clustered indexes, 124–128
- extents, 123
- heaps, 123
- nonclustered indexes on clustered tables, 130–131
- nonclustered indexes on heaps, 128–130
- overview, 122
- pages, 122–123

TABLESAMPLE clause, 177–180

terminology

- acyclic graphs, 460
- connected graphs, 460
- directed acyclic graph (DAG), 460
- directed graphs, 460
- forests, 461
- graphs, 460
- hierarchies, 461
- National Institute of Standards and Technology (NIST), 460
- resource for formal definitions, 460
- rooted trees, 461
- trees, 461
- undirected graphs, 460

textual showplans, 48–50, 116–117

tiebreakers

- aggregations, 319–321
- APPLY table operator. *See* APPLY table operator
- correlated subqueries, 196–199

nonunique sort column for row numbers, 229–230

TOP option. *See* TOP option

tools for query tuning

- analyzing execution plans overview, 107–108
- clearing cache, 105
- Database Engine Tuning Advisor (DTA), 121
- dynamic management objects, 106
- graphical execution plans, 108–115
- hints, 119–121
- measuring run time of queries, 106–107
- overview, 105
- Profiler, 121
- STATISTICS IO session option, 106
- syscacheobjects, 105
- textual showplans, 116–117
- tracing, 121
- XML showplans, 117–119

TOP option

- applying, 18–19
- DELETE statements, 385–388
- determinism, 383–384
- first page solution, 403–404
- input expressions, 385
- INSERT statements, 385–388
- matching current and previous occurrences solution 1, 398–399
- matching current and previous occurrences solution 2, 399–400
- matching current and previous occurrences solution 3, 400–401
- matching current and previous occurrences solution 4, 401–402
- matching current and previous occurrences solution overview, 397–398
- median value solution, 413–415
- modifications, 385
- n rows for each group solution 1, 392–394
- n rows for each group solution 2, 394
- n rows for each group solution 3, 395

- n rows for each group solution 4, 395–396
- n rows for each group solution 5, 396–397
- n rows for each group solution overview, 391–392
- next page solution, 404–409
- OR operator, 405–408
- ORDER BY clause, 383–384
- overview, 4, 381
- paging solution overview, 402–403
- previous page solution, 409–411
- random rows solution, 411–412
- SELECT TOP option basic example, 382
- SELECT TOP option overview, 381–382
- SELECT TOP PERCENT option example, 382–383
- SET ROW COUNT option, 385
- WITH TIES option, 384
- UPDATE statements, 385–388
- trace performance workload, 85–89
- tracing, 121
- transitive closure
 - creating script for fn_BOMTC (UDF solution), 533–534
 - creating script for fn_RoadsTC (UDF solution), 539, 545–546
 - generating all paths in BOM, 534–535
 - generating closure of roads, 538
 - isolating shortest paths in BOM (CTE solution), 535–537
 - loading shortest road paths into tables, 545–546
 - overview, 530–531
 - returning all paths and distances in roads, 541
 - returning shortest paths in roads, 541–545
 - running code for BOM (DAG), 531–533
- undirected cyclic graphs overview, 537
- trees
 - compared to DAG, 462
 - employee organizational chart scenario, 462–464

- forests, 461
- graphs. *See* graphs
- iteration. *See* iteration
- nested sets. *See* nested sets
- overview, 459–460, 461
- recursion. *See* recursion
- resource for formal definitions, 460
- rooted trees, 461
- trivial plan optimization, 41
- TRUNCATE TABLE vs. DELETE statement, 435
- tuning
 - index. *See* index tuning
 - query. *See* query tuning
- type derivation, 39

U

- UDA. *See* user-defined aggregates (UDA)
- uncommon predicates, 209–211
- undirected graphs
 - creating script for fn_RoadsTC (UDF solution), 539, 546–548
 - generating transitive closure of roads, 538
 - loading shortest road paths into tables, 545–546
 - overview, 460, 537
 - returning all paths and distances in roads, 541
 - returning shortest paths in roads, 541–545
 - road system scenario, 468–471
- UNION operator
 - flattening with algebraizer, 38
 - overview, 304
 - UNION ALL set operator, 304
 - UNION DISTINCT set operator, 304
- unique sort column, 228–229
- unordered clustered index scans, 132–134
- unordered covering nonclustered index scans, 134–136, 157
- unordered nonclustered index scan + lookups, 144–148, 158

594 UNPIVOT table operator

UNPIVOT table operator, 24–27

unpivoting vs. pivoting, 341–343

update plans

Assert operator, 59

cost strategies, 60

creating clustered index, 62–63

execution plan for update plans, 61–62

Halloween spool, 61, 63

maintaining indexes, 59

overview, 59

per-row and per-index, 60–61

performance, 60

stages, 59

UPDATE statements

Assert operator, 59

assignment UPDATE statements, 452–454

assignments overview, 450

cost strategies, 60

creating clustered index, 62–63

execution plan, 61–62

Halloween spool, 61, 63

joins, 443–447

maintaining indexes, 59

OUTPUT clause, 447–449

per-row and per-index plans, 60–61

performance, 60

read and write stages, 59

TOP option, 385–388

updates for SQL Server, xxiv

updating data

assignment SELECT statements, 450–452

assignment UPDATE statements, 452–454

overview, 443

performance considerations, 454–457

SELECT statement assignments overview, 450

UPDATE statement assignments overview, 450

UPDATE statements using joins, 443–447

UPDATE statements with OUTPUT clause,
447–449

user-defined aggregates (UDA)

C# code, 348

CLR code in databases, 347–348

creating assemblies in Visual Studio 2005,
353–357

enabling CLR and querying catalog views, 357

implementing, 348

overview, 347

testing, 357–358

Visual Basic .NET code for UDA, 351

user-defined function (UDF)

ancestors, creating fn_managers function,
484–485

ancestors, testing fn_managers function,
485–486

nested sets, creating script for
fn_empsnestedsets function, 523

nested sets, materializing relationships in
tables, 526

nested sets, testing script for
fn_empsnestedsets function, 525

overview, 472

subordinates, creating fn_subordinates1
function, 472–474

subordinates, getting other subordinate
attributes, 474–475

subordinates, testing fn_subordinates1
function, 474

transitive closure, creating script for
fn_BOMTC, 533–534

transitive closure, creating script for
fn_RoadsTC, 539

V-W

vertical vs. horizontal operations, 303

views, indexed, 153–155

wait times

analyzing at instance level, 71–79

correlating with queues, 80–81

WHERE filter
 applying, 11–12
 overview, 4
WITH TIES option, 384

X-Y

XML showplans, 50–51, 117–119
XOR operator, 364
Year-To-Date (YTD) aggregations, 330–331

