

Symbols

_id (primary key field) 15, 25
 _id field, for replica set members 171
 . (dot operator), with queries 85
 \$atomic (to prevent yielding) 124
 \$box (spatial query selector) 277
 \$center (spatial query selector) 277
 \$centerSphere (spatial query selector) 278
 \$cmd collection 34
 \$cmd.sys.inprog (virtual collection for current op) 231
 \$cmd.sys.unlock (unlock command path) 235
 \$elemMatch (query operator) 87–88
 \$in (query operator) 61
 \$maxElement (explain field) 149
 \$maxKey 197
 \$minElement (explain field) 149
 \$minKey 197
 \$mod (query operator) 90
 \$natural (sort modifier), for querying the oplog 166
 \$natural (sort operator) 70, 146
 \$near (spatial index query selector) 275

\$nearSphere (spatial query selector) 278
 \$regex (query operator) 90
 \$slice (projection operator) 91
 \$type (query operator) 72
 \$unset 27
 \$within (spatial query selector) 277

Numerics

10gen 5
 subscription services 5
 32-bit architecture 21
 32-bit integer 72
 64-bit architecture 21
 64-bit integer 72

A

address already in use (error message) 246
 addshard command 193, 213
 addShard() method 193
 addUser() method 227
 administration 33, 35
 aggregation 92–95
 calculating averages 93
 finalizer function 93
 AGPL (GNU) 13
 allPlans (explain output) 151
 Amazon EC2 186, 222
 analytics 257–258
 production use case 20
 schema design for 257
 shard key for 208

and (Boolean) 81, 83
 anti-patterns 258–259
 bucket collections 258
 careless indexing 258
 collection per user 259
 large documents 258
 motley types 258
 unshardable collections 259
 Apache license 15
 AppEngine 5
 application design
 dynamic attributes 48
 Twitter archiver 47, 52
 arbiterOnly 171
 arbiters 159
 ArchLinux 242
 arrays
 indexing 87
 See also multikey indexes
 querying 80, 86–88
 update operators for 121–122
 using dot notation with 87
 atime (file system attribute) 221
 atomic operations, with targeted updates 104
 atomicity 109
 auth (mongod option) 227
 auth() method 227
 authentication 226–228
 read-only access 227
 averages, calculating manually 105

B

- backups 234–235
 - for sharded clusters 215–216
 - locking the database for 235
- balancer process
 - sh.isBalancerRunning()
 - method 216
 - sh.setBalancerState()
 - method 216
 - stopping 215
- balancing
 - balancer process 190
 - triggering of 190
- big-endian support 219
- BigTable (Google's internal data store) 19
- binary data
 - BSON binary subtypes 261
 - storage of 260–262
- bind_ip (mongod option) 226
- BLOB (RDBMS type) 260
- BSON 44–45
 - :Binary (Ruby class) 261
 - :OrderedHash (Ruby class) 86
 - custom types 73
 - examining raw 233
 - internal format example 44
 - key name overhead 71
 - maximum size of 177
 - numeric types 72
 - Object ID type. *See* Object IDs
 - preserving key order 86
 - serialization of 70–72
 - sizing and updates 125
 - strings 72
 - time types 73
 - timestamp type 164–165
 - types 72–73
 - valid key names for 71
- BSON types, min key and max key 197
- bsondump utility 233
- BSONObjBuilder (C++ class) 271
- B-trees 136–137
 - advantages of 136
 - estimating storage size 137
 - maximum key size 137
 - node structure 137
- buildIndexes (replica set option) 172
- bulk loading data 74

C

- C++ driver 270–273
 - as an introduction to the source code 270
 - creating connections with 272
 - creating documents with 270
 - sample MongoDB program 272
- capacity planning 211
- capped collections 68–70
 - and replication 164
 - for implementing a queue 255
 - indexes and 69
 - limitations of 70
 - natural ordering of 70
- Cassandra database 17
- CentOS 242
- changelog (config collection) 199
- character encoding 72
- chunks 189
 - collection storing chunk ranges 196
 - counting 196
 - default max size 190
 - logical vs. physical 189
 - pre-splitting for faster distribution 212
 - problem when too large to split 206–207
 - splitting and migrating of 190
- clock skew 222
- cloud deployment 222
- collection scans 83, 148
- collections 67–70
 - automatic creation of 25
 - capped. *See* capped collections
 - drop() method 28
 - listing 33
 - renaming 68
 - sharding existing 211
 - stats 34
 - system. *See* system collections
 - valid names for 67
 - virtual namespacing of 68
- collstats command 35
- command shell 14, 24
 - getting help 35

- command-line options
 - 246–247
 - getting from a running mongod 247
- commands 34
 - implementation of 34
 - max return size 96
 - runCommand() method 35
- compensation-driven mechanisms 257
- compiling from source 245
- concurrency 124–125, 220
 - optimistic locking 104
 - yielding strategy 124
- config database 196
- config files 247
- config servers 188
 - deployment of 209
 - failure of 217
 - two-phase commit and 188
- configdb (mongos option) 193
- configsvr (mongod option) 192
- connecting to mongod 39
- connection URIs 267
- CouchDB (Apache document-based data store) 19
- count 78
- count command 26
- covering indexes 154–155
- CPU. *See* hardware requirements
- createCollection() method 67
- CSV format 225
- currentOp() method 142, 230
- cursor types
 - BasicCursor 31
 - BtreeCursor 32
- cursors 40
 - BasicCursor (explain output) 148
 - BtreeCursor (explain output) 148
 - iterating 41
 - reasons for 40

D

- data center awareness 182
- data centers
 - multiple with replica sets 176
 - multiple with sharding 210
- data directory 242

data files 66
 .ns file 66
 allocation of 66
 copying 235
 limiting the size of 66
 data modeling. *See* schema design
 data types, BSON definition of 44
 databases 65
 allocating initial data files 25
 automatic creation of 25
 creating 25
 listing 33
 stats 33
 databases (sharding metadata collection) 214
 db.isMaster() method 160
 DBClientReplicaSet (C++ class) 272
 dbpath (mongod option) 246
 dbpath does not exit (error message) 245
 dbstats command 34, 67
 deleteIndexes command 140
 deletes 42, 124
 denormalization, benefits of 63
 deployment 219–228
 file systems 221
 in the cloud 222
 server configuration 223
 design patterns. *See* schema design
 dictionary (Python primitive) 15
 directoryperdb (mongod option) 221
 discover (mongostat option) 231
 disks 220
 diagnosing performance of 237
 RAID configurations 221
 suggestions for improving performance of 239
 distinct (aggregation function) 96
 document model 4
 documents 5, 13
 advantages of 4, 7
 deep nesting of 74
 example social news site entry 5
 lack of enforced schema 7
 language representation of 39

 relation to agile development 7
 size limits 74
 size vs. performance considerations 74
 space overhead 71
 versioning 63
 dot notation
 ambivalence to arrays and sub-documents 87
 using with arrays 120
 dot-operator, in a projection 91
 double (numeric data type) 72
 drivers 15, 43–46
 "fire and forget" behavior 46
 API design 38
 functions performed 43
 networking protocol 45–46
 object id generation 43
 replica set failover and 179
 replication and 177
 safe mode. *See* getLastError command
 write concern and. *See* write concern
 drop (mongorestore option) 234
 dropDups (option for unique indexes) 138
 dropIndex() method 141
 duplicate key error 138
 durability, trade-off with speed 11
 dynamic attributes 255–256
 dynamic queries 8
 Dynamo 18

E

EBS (elastic block storage) 222
 EC2 12
 See also Amazon EC2
 e-commerce 7
 RDMBS suitability 57
 sample product document 58
 schema for products and categories 58
 embedding vs. referencing 249
 enablesharding command 194
 endianness 219
 ensureIndex() method 140
 entity-attribute-value pattern 7

error messages 245
 eventual consistency 17, 159
 excluding fields from query results 90
 explain 147–149
 millis value 147
 n value 31
 nscanned value 31, 147
 output of 147
 viewing attempted query plans 151
 exporting data 225
 ext4 (file system) 221

F

f (mongod config file option) 247
 failover
 and replication 158
 example of 162
 Fedora 242
 file descriptors 221
 file systems 221
 finalize function
 as used with group 97
 as used with map-reduce 99
 find method 77
 find_one method 77
 findAndModify
 command 112–118
 for implementing a queue 254
 implementing transactional semantics with 114
 options for 123
 fire-and-forget (default write mode) 11
 foreign keys. *See* relationships
 fork (mongod option) 246
 Foursquare (location-based social network) 274
 FreeBSD 242
 fs.chunks (GridFS collection) 263
 fs.files (GridFS collection) 263
 fsync command 235

G

gem (Ruby command) 247
 genOID() function (C++) 271
 geoNear command 276
 geospatial indexing. *See* spatial indexing

getCmdLineOpts
 command 247

getIndexKeys() method 151

getIndexSpecs() method 141

getLastError command 46, 179

 j (sync journal) option 180

 specifying replication
 defaults 172

getLastErrorDefaults (replica
 set option) 172

getLastErrorModes 183

getReplicationInfo()
 method 166

getSiblingDB() method 193

Gizzard (manual sharding
 framework) 186

GridFS 262–265

 collections used by 263

 comparison with standard
 file systems 262

 default chunk size 262

 using from mongofiles 265

 using from Ruby 262, 264

GridIO (Ruby GridFS
 class) 264

group command 92, 94

 aggregation function 96

 command options 96–97

 limits of 96

H

halted replication 166

hardware requirements
 219–223

 CPU 219

 disks 220

hash (Ruby primitive) 15

help() method 35

hidden (replica set
 option) 172

hierarchical data. *See* trees

hint (forcing an index) 152

Homebrew 243

horizontal scaling 12

host, replica set config
 option 171

HostAndPort (C++ class) 272

HTTP. *See* REST interface

I

importing data 225

IN (SQL directive) 61

index types

 compound-key indexes
 133–135

 difference between single
 and compound-key
 133–134

 multikey indexes 139–140

 single-key indexes 133

 sparse indexes 138–139

 unique indexes 138

indexBounds (explain
 field) 149

indexes

 backups for 143

 building in the
 background 143

 building process 141

 compaction of 144

 creating 59

 creating and deleting 140

 maximum key size 137

 when to declare them 141

 write lock when building 143

indexing 10

 administration 140–144

 arrays. *See* multikey indexes

 B-tree (data structure).
 See B-trees

 caution about building
 online 142

 compound-key indexes.
 See compound-key indexes

 cookbook analogy 10,
 130–133

 efficiency issues 135–136

 ensureIndex() method 31

 geospatial. *See* spatial indexing

 getIndex() method 31

 importance of number of
 documents scanned 134

 null values and 138

 offline 143

 ordering of keys 135

 performance cost 135

 RAM requirements 135

 sharding and 204–205

 unique indexes 59, 63

See also query optimization

index-only queries. *See* cover-
 ing indexes

inserts 40

 bulk 74

 max insert size 75

 safe 46

 with unique indexes 60

installing

 on Linux 241–242

 on OS X 242–243

 on Windows 244

 with Linux package
 managers 242

 with OS X package
 managers 243

iostat utility 237

isMaster command 177

it (iterate) shell macro 30

J

j (write concern option) 224

Java driver 268–270

 connection options 269

 creating documents with 268

 sample MongoDB
 program 269

 using WriteConcern
 objects 269

JavaScript 24

 querying with 88–89

 this (keyword) 88

 when to use 88

JavaScript shell. *See* command
 shell

join tables 250

joins 58

 alternatives to 64, 80

 client-side 80

 complexity introduced 4

 complexity of 8

 in RDBMS 60

journal (data directory) 224

journaling 11, 223–225

 consequences of
 disabling 12

 guarantees provided by 224

 j (getLastError option) 180

 relationship to
 replication 157

JSON 25

 for document
 representation 39

K

keyFile (mongod option) 228

keys names, saving space by
 shortening 86

key-value stores 17, 56

 implementing secondary
 indexes with 9

key-value stores (*continued*)
 query model 9
 use cases 17
 killOp() method 231

L

licensing, core server 13
 limit 78, 92
 LinkedHashMap (Java class) 15
 listDatabases command 42
 listshards command 193
 little-endian support 219
 load balancing, with replication 158
 local database 164
 locality 258
 definition of 206
 in choice of shard key 206
 locking the database against writes 235
 locking. *See* concurrency
 logappend (mongod option) 229
 logging 228
 slow queries 144
 logout command 227
 logpath (mongod option) 191, 229, 246
 logrotate command 229
 long polling 166
 LVM (logical volume manager) 221

M

MacPorts 243
 maintenance, with replication 158
 many-to-many. *See* relationships
 map-reduce 94–95, 98–99
 emit() function 94
 map function 94
 querying the results collection 95
 reduce() function 95
 using iteratively 99
 master-slave replication 177
 disadvantages of 177
 max (finding max value) 95
 maxBsonObjectSize field (ismaster command) 177
 MD5 (as shard key) 206

MD5 (storage of) 261
 Memcached 17
 memory architecture 219
 min (finding min values) 95
 mmap (system call) 14
 mongo (executable) 14, 24
 mongod 14
 mongod (executable) 14
 mongod.lock (lock file) 66, 246
 MongoDB
 definition of 4
 design philosophy 16
 document-oriented data model 5
 open source status 5
 operating system support 13
 reasons for using 16–21
 suitability for web applications 5
 uniqueness of data model 4
 with object-oriented languages 4
 mongodump utility 16, 234
 mongoexport utility 16, 225
 mongofiles utility 265
 mongoimport utility 16, 21, 225
 mongorestore utility 16, 234
 mongos 187
 failure of 217
 using mongodump with 215
 mongos (executable) 14
 mongosniff utility 16, 233
 mongostat utility 16, 231
 monitoring 228–233
 custom plug-ins for 233
 moveChunk command 212
 moveprimary command 214
 multikey indexes 48
See also index types
 multi-updates. *See* updates
 Munin 232
 MySQL 17, 19
 transaction logging 11

N

Nagios 232
 namespaces 66
 NASDAQ (example data set) 145
 noprealloc (server option) 66
 normalization 3, 6

NoSQL 4
 nssize (server option) 66
 NTP (network time protocol) 222
 NumberInt() method 72
 NumberLong() method 72

O

Object IDs 15, 40
 as shard keys 205
 compound 85
 encoded creation timestamp 44
 format 43
 string vs. binary encoding 45
 use arrays of with \$in queries 81
 object mappers
 purpose of 57
 using with MongoDB 58
 object-relational mappers (ORMs) 15, 20
See also object mappers
 one-to-many. *See* relationships
 open 13
 oplog 164–166
 default size of 167
 how operations are logged 165
 idempotence of operations 166
 querying manually 164
 sizing of 167
 structure of entries 165
 oplog.rs (system collection) 70, 164
 optimistic locking.
See concurrency
 or (Boolean) 81, 83
 or (logical), expressed using \$in 83
 Oracle database 17

P

padding factor 126
 page faults 135
 pagination 78
 optimization of 92
 partitioning. *See* sharding
 performance
 troubleshooting 237–240
 permission denied (error message) 245

PHP driver 266–268
 connection options 267
 creating documents with 266
 persistent connections
 and 267
 sample MongoDB
 program 267
 PNUITS (Yahoo's internal data
 store) 19
 port (mongod option) 246
 positional operator. *See* update
 operators
 PostGIS 274
 PostgreSQL 17
 precomputation 258
 prefix queries 253
 primary keys. *See* `_id` field
 priority (replica set
 option) 171
 production deployments, *The
 Business Insider* 20
 Project Voldemort 17
 projections 79–80, 90–91

Q

queries 26
`_id` lookups 77
`.` (dot operator) 80
 against arrays 80
 against null values 79
 Boolean operators 83–85
`explain()` method 31
 matching a prefix 80
 matching sub-
 documents 85–86
 object id reference
 lookups 77
 range 30, 80
 range type matching 82
 ranges 81–82
 regular expressions 80
 selecting a subset of fields.
See projections
 set operators 82–83
 with complex types 86
 query operators 80–92
`$all` (universal inclusion
 operator) 82–83
`$and` (Boolean AND) 83
`$exists` (attribute
 existence) 83
`$gt` (greater than) 9, 30,
 80–81

`$gte` (greater than or equal
 to) 81
`$in` (array inclusion) 81
`$in` (existential inclusion
 operator) 82
`$lt` (less than) 30, 81
`$lte` (less than or equal to) 81
`$mod` (modulus) 90
`$ne` (attribute not equal
 to) 83
`$ne` (not equal to) 109
`$nin` (negative existential
 inclusion operator) 82–83
`$not` (negation) 84
`$not` (selector negation) 83
`$or` (selector Boolean
 OR) 83
`$or` (selector logical OR) 84
`$regex` 90
`$size` (array operator) 88
 combining for a single
 key 82
 query optimization 144–155
 common query patterns and
 indexes for 153–155
`explain()` method. *See* explain
 profiling queries. *See* query
 profiler
 with compound-key
 indexes 154
 with single-key indexes
 153–154
 query optimizer
 caching and expiring query
 plans 153
 internal 149–153
 running queries in
 parallel 151
 query optimizing with covering
 indexes. *See* covering
 indexes
 query options 90
 query profiler 146–147
 query selectors 26, 81, 90
 queues (implementing)
 254–255

R

RAM
 determining working set
 size 238
 hardware requirement 220
 in-memory databases 11
 page size 135

range queries, optimizing
 indexes for 154
 read (Ruby driver connection
 option) 181
 read scaling 181
 consistency and 181
 limitations of 181
 real-time analytics. *See* analytics
 recovery
 from categorical node
 failures 174
 from network partitions 174
 with a complete resync 174
 reduce functions 92
 redundancy, provided by
 replication 158
 regular expressions
 querying with 89–90
 with the `$not` operator 84
 reIndex command 144
 reIndex() method 236
 relationships
 indexes for 78
 many-to-many 60–61,
 250–251
 one-to-many 62, 249–250
 self-joining. *See* trees
 releases 13
 versioning of 241
 removeshard command
 213–214
 removeUser() method 227
 removing data, remove()
 method 28
 renameCollection()
 method 68
 repair (mongod option) 235
 repairDatabase command 235
 replica sets 159–177
 administration of 169–177
 and automated failover 10
 authentication and 228
 commit and rollback
 168–169
 config document 169
 configuration
 document 169–170
 configuration options
 171–173
 connecting to 177–179
 failover and recovery of
 174–175
 forcing reconfiguration
 of 176
 getting the status of 161
 heartbeat 167

- replica sets (*continued*)
 - how failover works 168
 - minimum recommended configuration 159
 - production
 - configurations 175–177
 - reconfiguring 174
 - re-election after failover 163
 - setting up 159–161
 - table of possible states 173
 - tagging 172, 182
 - with multiple data centers 176
- replication 10, 157–158
 - clock skew and 222
 - failover and 158
 - failure modes it protects against 157
 - how it works 163–169
 - journaling and 157
 - use cases for 158–159
- replset.minvalid (system collection) 164
- ReplSetConnection (Ruby connection class) 178
- replSetGetStatus
 - command 173
- rest (mongod option) 246
- REST interface 232, 246
- Riak 17
- rollback
 - implementation 117
 - implementing with findAndModify 116
 - See also* replica sets
- rs.reconfig() method 174
- rs.status() method 161, 173
- Ruby
 - commands 42
 - connecting to MongoDB 39
 - ursors 40–41
 - deletes 41
 - hashes in 1.8, ordering 86
 - inserting documents with 40
 - installing 247–248
 - installing the driver 38
 - introduction to 38
 - irb (REPL) 39
 - querying the database 40
 - representing MongoDB documents 39
 - sample application 47–52
 - Time objects 73
 - updates 41
- RubyGems 38
- S**

 - safe mode 11, 179
 - Sagas paper 257
 - scalability, as an original design goal 5
 - scaling
 - for reads. *See* read scaling
 - strategies for 239
 - See also* sharding
 - scanAndOrder (explain field) 148
 - schema design
 - denormalization 61, 64
 - dependence on database features 56
 - dynamic attributes 58, 60
 - e-commerce examples. *See* e-commerce
 - embedded documents. *See* sub-documents
 - exceptions to the standard patterns of 56
 - for RDBMSs 56
 - FriendFeed example 56
 - modeling tags 60
 - patterns 249, 258
 - principles 56–57
 - referencing other documents 64
 - relationships. *See* relationships
 - relevance of application access patterns 57
 - using large documents 74
 - using meaningful ids (slugs) 59
 - using sub-documents for dynamic attributes. *See* sub-documents
 - security 226–228
 - authentication. *See* authentication
 - JavaScript injection 89
 - sequential vs. random writes 11
 - SerialServer (sharded query type) 201
 - serverStatus command 229–230
 - settings (config metadata collection) 215
 - sh (sharding shell helper object) 193
 - sh.enableSharding() method 194
 - sh.moveChunk() method 212
 - sh.shardCollection() method 194
 - sh.splitAt() method 212
 - sh.status() method 193, 198
 - shard clusters
 - adding a shard 213
 - backing up 215–216
 - bypassing the balancer 213
 - checking chunk distribution 198
 - failover and recovery of 216–217
 - importance of
 - monitoring 213
 - monitoring of 212
 - network communication requirements 226
 - querying and indexing 200–205
 - querying the change log 199
 - removing a shard 213
 - unsharding a collection 214
 - shard keys 189
 - coarse-grained keys 206
 - examples of 194
 - ideal attributes of 207
 - properties of an ineffective shard key 205, 207
 - random distribution 206
 - shardcollection command 194
 - sharding 12, 185, 208–217
 - across data centers 210
 - checking which collections are sharded 194
 - displaying configuration and status 193
 - estimating cluster size 211
 - handling data center failure 211
 - how it works 187–190
 - manual
 - implementations 185
 - problem definition 185–186
 - processes required 191
 - production deployment techniques 208–217
 - query types 200
 - rationale for distributing collections 188
 - sample deployment topologies 209–210
 - shard keys 189
 - when to use 186
 - shards (composition of). *See* sharding

shardsvr (mongod option) 191
 shell
 examining a method's implementation 36
 numeric representations 72
 tab completion 35
 single point of failure 12
 skip 78, 92
 optimizing 92
 slave_ok (connection parameter) 178
 slaveDelay (replica set option) 172
 slaves (system collection) 164
 slowms (mongod flag) 145
 smallfiles (server option) 66
 solid state drives 186
 sorting 78, 91–92
 optimizing indexes for 153–154
 optimizing skips with 92
 to find maxima and minima 95
 spatial indexes
 center queries 277
 creating 275
 querying with 275
 querying within a box 277
 with compound keys 277
 with spherical geometry 278
 spatial indexing 274, 278
 split command 212
 splitting, algorithm with small data sizes 199
 SPOF. *See* single point of failure
 SQL 8–9, 24
 LIKE directive 80
 max() function 95
 min() function 95
 range operators 81
 SSDs. *See* solid state drives
 stale data. *See* halted replication
 state machines 112, 114
 stats() methods.
 See collections
 sub-documents 62
 updating 108
 vs. top-level documents. *See* embedding vs. referencing
 subtypes. *See* binary data
 system collections 70
 system.indexes 33
 system.indexes (system collection) 70, 140

system.namespaces
 collection 70
 system.profile (system collection) 146
 system.replset (system collection) 164
 system.users (system collection) 227

T

table scans. *See* collection scans
 tables, comparison with MongoDB documents 58
 tagging
 in replica sets 182
 setting
 getLastErrorModes 183
 TCP sockets 43
 this (JavaScript keyword), map-reduce usage 94
 thrashing (disk) 136
 thumbnails (storage of) 260
 time
 as represented in object IDs 43
 storing 73
 storing time zones 73
 Timestamp() (JavaScript constructor) 165
 timestamps, in the oplog 165
 top command 230
 transaction logging.
 See journaling
 transactional semantics 112–114
 adding an item to a shopping cart 110
 inventory management 112, 118
 voting on reviews 108
 transactions, implementing in MongoDB 256–257
 trees 251–254
 category hierarchy
 example 105–108
 denormalized ancestor pattern 251
 materialized path pattern 251
 representing threaded comments with 251
 using an ancestor list 61
 troubleshooting
 installation 245–246

truncate. *See* removing data
 tuning 14
 Twitter, storing tweets 21

U

Ubuntu 242
 uniqueness (enforcing) 46
 unlocking the database (after fsync lock) 235
 update operators
 \$ (positional operator) 108, 122
 \$addToSet 28
 \$inc 104, 108, 119
 \$pullAll 117
 \$push 28, 103, 108
 \$rename 120
 \$set 26, 41, 102, 120
 \$sunset 120
 updates 26
 advantages of targeted updates 103
 arguments provided 26
 by replacement vs. by operator 102–104
 distinguishing syntax from query syntax 118
 findAndModify. *See* findAndModify command
 in-place 120, 125
 multiple documents 108, 119
 on multiple documents 41
 performance of 125–126
 targeted 26
 upgrading 236
 upserts 110–111, 119
 URIs. *See* connection URIs
 use cases 20
 agile development 20
 analytics and logging 20
 caching 21
 variable schemas 21
 web applications 20
 UTC (Coordinated Universal Time) 73
 UTF-8 (character encoding requirement) 72

V

v (mongod verbosity option) 229

versioning 13, 241
 See also releases
vertical scaling 12
virtual collections 250
virtual memory, monitoring 230
votes (replica set option) 171

W

w (write concern option) 179
web applications 3
web console 232
Windows 244
Windows Time Service 222
wire protocol 43
WordSquared (online
 game) 274
working data set 136
working set 159, 238
write concern 179–180
 implementation of 180
write lock. *See* concurrency

wtimeout (write concern
 option) 179

X

xfv (file system) 221

Y

yielding. *See* concurrency