Feature	Riak	Redis	Cassandra	MongoDB	Neo4J
Architecture	Peer-to-peer	Master-slave	Peer-to-peer	Replica set	Master-slave / Causal Cluster
Keywords	Gossip protocol, hinted handoff, Riak ring, siblings	Binary-safe keys, TTL, Sentinel	Tombstone, SSTable, quorum, memtable	Shard key, replica set, BSON	Traversals, ACID, causal consistency
Replication	N-value, hinted handoff, vector clocks	Master-slave, asynchronous	Tunable consistency, quorum-based	Replica sets with primary node	Causal consistency (enterprise)
Sharding	Riak ring with consistent hashing	Hash slots (16384 slots)	Consistent hashing, partition keys	Shard key	Manual (Enterprise supports clustering)
Consistency	BASE (Eventual, quorum-based: (R+W > N))	BASE (Strong eventual)	BASE (Tunable: `ALL`, `QUORUM`, `ONE`)	ACID (Multi-document configurable)	ACID
Writes	Vector clocks ensure version tracking	In-memory, replicated to slaves	Log-structured, SSTables, upserts	Journaled writes (WiredTiger engine)	Transactional for graphs
Reads	Tunable quorum, often slower due to eventual	Master-slave consistency	Tunable quorum, fast with eventual data	Primary or secondary nodes, configurable	Optimized for graph traversals
Deletes	Vector clocks track tombstoned versions	Deletes in-memory, expire via TTL	Tombstones in SSTables, cleared on compaction	Document deletion and journaling	Traversal-based deletions
Indexing	Limited (full-text search via Riak Search)	Key-based lookups, some range queries	Primary and secondary indexes	Compound and text indexes	Native graph index
Transactions	Not supported (quorum emulates consistency)	Atomic commands, MULTI/EXEC blocks	Lightweight transactions, conditional updates	Multi-document ACID	ACID transactions
Fault Tolerance	High (hinted handoff, gossip protocol)	Sentinel for failover, cluster redis	High with quorum and replication factor	High with replica set failover	High in enterprise editions
Persistence	BASE principles (quorum + eventual)	Snapshot-based (RDB) or AOF logs	SSTables and commit logs	Journaling and snapshots	Journaling
Performance	Moderate, optimized for fault tolerance	Very high for in-memory workloads	High for write-heavy workloads	Moderate for balanced workloads	Slower for very large graphs
Best Use Cases	Fault-tolerant key-value storage, metadata	Caching, ephemeral session storage	Write-heavy systems, time-series data	Semi-structured data, flexible schemas	Complex relationships and queries
Weaknesses	Conflict resolution, slower writes	Memory-limited, single-threaded	Read overhead, slower compactions	Single primary bottleneck without sharding	Limited sharding, scaling for huge datasets