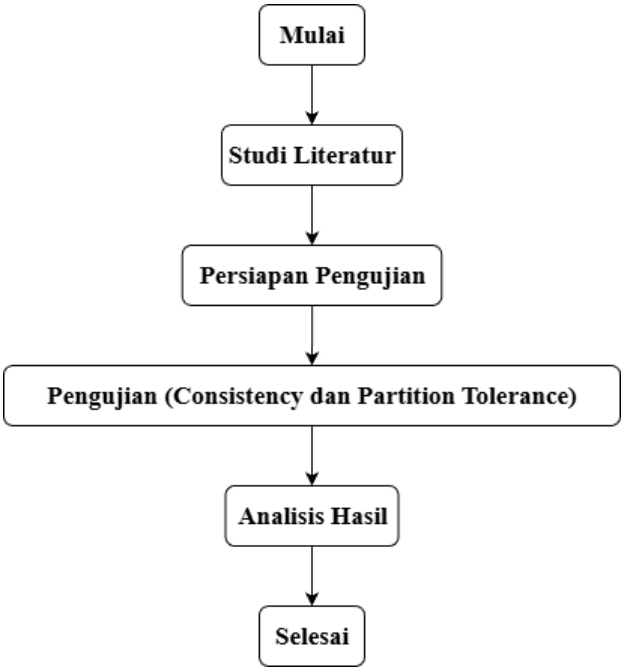
BAB III

METODE PENELITIAN

3.1 Tahapan Penelitian

Penelitian ini menggunakan pendekatan eksperimen untuk mengukur tingkat *consistency* dan *partition tolerance* pada sistem replikasi MongoDB. Tahapan penelitian mulai dari studi literatur, persiapan pengujian, pelaksanaan pengujian, hingga analisis hasil. Tahapan penelitian disajikan pada Gambar 3.1.



Gambar 3.1 Tahapan Penelitian

3.2 Studi Literatur

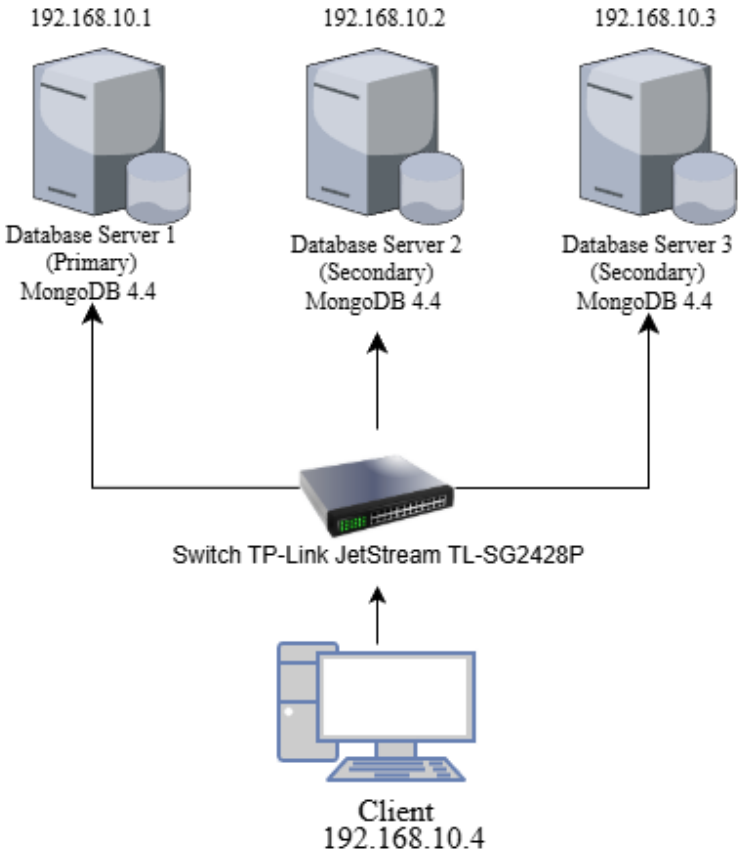
Tahapan studi literatur dilakukan untuk memperoleh pemahaman konseptual terkait NoSQL *database*, replikasi MongoDB, Teorema CAP, serta konsep *consistency* dan *partition tolerance*. Referensi diperoleh dari jurnal ilmiah dan

publikasi akademik yang relevan. Materi yang dikumpulkan digunakan sebagai dasar dalam merancang skenario pengujian serta menentukan parameter pengukuran, meliputi *replication lag*, *stale reads*, *majority write acknowledgement latency, request success rate*, *recovery time*, dan *rollback data*.

3.3 Persiapan Pengujian

3.3.1 Topologi Sistem

Pengujian dilakukan menggunakan jaringan lokal berbasis *switch* pada tiga *server* yang membentuk *replica set* dan satu *client* penguji. Topologi ditampilkan pada Gambar 3.2



Gambar 3.2 Topologi Replikasi

Gambar 3.2 menggambarkan topologi sistem yang digunakan pada pengujian. Satu komputer *client* terhubung ke *switch* dan digunakan untuk mengirim beban uji menggunakan *script Python* berbasis *driver pymongo*. Pada sisi *server* terdapat tiga node MongoDB, yaitu *Database Server* 1 sebagai *primary* serta *Database Server* 2 dan *Database Server* 3 sebagai *secondary*. *Primary* menerima seluruh operasi tulis, sedangkan *secondary* menarik data hasil replikasi melalui *operation log* (oplog) dari *primary*. Komunikasi antar node menggunakan protokol TCP pada port 27017 dan seluruh perangkat berada dalam satu jaringan lokal agar pengujian berlangsung stabil dan terkontrol. Simulasi partisi jaringan dilakukan secara logis menggunakan *iptables* untuk memblokir konektivitas antar node. Partisi yang diterapkan merupakan partisi parsial, yaitu pemutusan koneksi antara *primary* dan salah satu *secondary*, sementara *secondary* lainnya tetap terhubung sehingga *quorum* (2 dari 3 node) tetap terpenuhi. Dengan konfigurasi ini, sistem masih dapat menjalankan operasi tulis dengan *writeConcern* = "*majority*". Penggunaan tiga node dipilih karena merupakan konfigurasi minimal yang memungkinkan terbentuknya mekanisme *majority* dalam *replica set*.

3.3.2 Spesifikasi *Hardware*

Hardware yang digunakan dalam lingkungan pengujian beserta spesifikasinya disajikan pada Tabel 3.1.

Tabel 3.1 *Spesifikasi Hardware*

Node	Peran	OS	*IP Address*	CPU	RAM
Node 1	*Primary*	Ubuntu 20.04.5 LTS	192.168.10.1	Intel Celeron N5105	4 GB
Node 2	*Secondary*	Ubuntu 20.04.5 LTS	192.168.10.2	Intel Celeron N5105	4 GB
Node 3	*Secondary*	Ubuntu 20.04.5 LTS	192.168.10.3	Intel Celeron N5105	4 GB
Client	Pengirim *Query*	Windows 11 Pro	192.168.10.4	Intel Core i5-8250U	4 GB

Tabel 3.1 menunjukkan bahwa *server* pengujian menggunakan spesifikasi yang sepadan (*homogen*), yaitu prosesor Intel Celeron N5105 dengan RAM 4 GB pada masing-masing *node*. Keseragaman spesifikasi perangkat diperlukan untuk memastikan bahwa perbedaan kinerja yang dihasilkan benar-benar disebabkan oleh beban sistem, bukan oleh variasi kapasitas perangkat keras. *Client* pengujian menggunakan prosesor Intel Core i5 karena membutuhkan kemampuan komputasi yang lebih tinggi untuk menjalankan proses pembangkitan *dataset*, eksekusi *script* beban, serta pencatatan data *secara real-time*. Perangkat *switch* yang digunakan mendukung teknologi Gigabit Ethernet sehingga mampu menyediakan latensi jaringan yang rendah dan stabil selama proses pengujian berlangsung.

3.3.3 Spesifikasi *Software*

Software yang digunakan dalam lingkungan pengujian beserta spesifikasinya disajikan pada Tabel 3.2

Tabel 3.2 *Spesifikasi Software*

Komponen	Spesifikasi
Database	MongoDB 4.4.29 (Replica Set)
Format Data	JSON/BSON
Bahasa Pemrograman	Python 3.10
Library	pymongo, csv, subprocess, datetime
Tools Partisi	Iptables
Sinkronisasi Waktu	Timestamp Python (UTC)

Tabel 3.2 memperlihatkan *Software* yang digunakan pada *server* dan *client*. Versi MongoDB yang digunakan adalah 4.4.29, karena versi ini mendukung *tunable consistency*, *automatic failover*, serta kompatibel dengan *driver* Python terbaru. Library pymongo memungkinkan pencatatan waktu secara presisi pada setiap proses penulisan maupun pembacaan data. *Tool iptables* digunakan untuk mensimulasikan kondisi partisi parsial secara terprogram, yaitu dengan memutus konektivitas antara primary dan salah satu secondary, sementara satu secondary lainnya tetap terhubung sehingga quorum tetap terpenuhi. Secara terprogram tanpa cabut kabel fisik, sehingga metodologi ini sesuai dengan standar eksperimen replikasi modern. Format data JSON/BSON dipilih karena merupakan struktur bawaan MongoDB dan lazim digunakan dalam sistem NoSQL.

3.3.4 Konfigurasi *Replica Set*

Konfigurasi replikasi dilakukan pada tiga server MongoDB dengan membentuk sebuah *replica set* yang terdiri atas satu primary node dan dua *secondary node*. Inisialisasi *replica set* dilakukan melalui terminal pada *primary node* menggunakan perintah *rs.initiate*() sebagai berikut.

```
      rs.initiate({
        _id: "rs0",
        members: [
          { _id: 0, host: "192.168.10.1:27017" },
          { _id: 1, host: "192.168.10.2:27017" },
          { _id: 2, host: "192.168.10.3:27017" }
        ]
})
```

Perintah tersebut mendeklarasikan nama *replica set* serta alamat setiap anggota *node* yang terlibat dalam mekanisme replikasi. Setelah proses inisialisasi selesai, sistem secara otomatis melakukan proses *election* untuk menentukan *primary*, sementara dua *node* lainnya berperan sebagai *secondary*. Pemilihan konfigurasi tiga *node* dilakukan karena merupakan jumlah minimal yang memungkinkan terbentuknya mekanisme *quorum* (mayoritas). Pada topologi ini, operasi tulis dengan *writeConcern = "majority"* mensyaratkan minimal dua *node* memberikan pengakuan sebelum operasi dinyatakan berhasil.

Dalam penelitian ini, setiap operasi tulis dikonfigurasikan menggunakan *writeConcern = { w: "majority", j: true, wtimeout: 5000 }*. Konfigurasi tersebut memastikan bahwa operasi tulis dianggap berhasil hanya setelah diakui oleh mayoritas anggota *replica set*.

Penelitian ini menggunakan dua konfigurasi koneksi yang berbeda sesuai dengan tujuan masing-masing metrik. Pada pengukuran *latency strong* sebagai representasi *consistency* kuat, koneksi dikonfigurasikan dengan *readPreference = "primary", readConcern = "majority",* dan *writeConcern = "majority"*. Konfigurasi ini memastikan bahwa proses pembacaan dilakukan langsung dari *primary* dan hanya terhadap data yang telah dikomit oleh mayoritas anggota

replica set. Dengan demikian, latensi yang diukur merepresentasikan waktu yang dibutuhkan sistem untuk menjamin konsistensi berbasis quorum.

Pada pengukuran *replication lag* dan *stale reads*, koneksi dikonfigurasikan dengan *readPreference = "secondary"* dan *readConcern = "local",* sementara operasi tulis tetap menggunakan *writeConcern = "majority"*. Pengaturan ini memungkinkan pembacaan langsung dari *secondary* tanpa menunggu komit mayoritas, sehingga keterlambatan propagasi replikasi dapat diamati secara operasional.

Perbedaan konfigurasi ini dirangkum pada Tabel berikut untuk memperjelas hubungan antara metrik dan parameter yang digunakan.

Tabel 3.3 Konfigurasi Parameter per Metrik

Metrik	*Node Target*	*readPreference*	*readConcern*	*writeConcern*
Replication Lag	*Secondary*	*Secondary*	*local*	*Majority*
Stale Reads	*Secondary*	*Secondary*	*local*	*Majority*
Latency Strong	*Primary*	*Primary*	*majority*	*Majority*
Request Success Rate	*Primary*	*primary*	*majority*	*Majority*
Recovery Time	*Replica Set*	*primary*	*majority*	*Majority*
Rollback Data	*Secondary*	*Secondary*	*local*	*1 (khusus uji rollback)*

Tabel 3.3 menjelaskan pembagian skenario pengujian berdasarkan kondisi jaringan yang diterapkan pada MongoDB *Replica Set* tiga node. Pengujian dibagi menjadi dua kondisi utama, yaitu kondisi normal dan kondisi partisi parsial, dengan tujuan untuk membandingkan karakteristik sistem dalam keadaan tanpa gangguan dan saat terjadi gangguan komunikasi sebagian.

Pada kondisi normal, seluruh node dalam *replica set* saling terhubung tanpa adanya pemutusan konektivitas jaringan. Dalam kondisi ini, mekanisme replikasi berjalan sebagaimana mestinya dan *quorum* selalu terpenuhi. Skenario ini digunakan untuk mengukur karakteristik *consistency* sistem, yang meliputi *replication lag, stale reads*, dan *latency strong*. Pengujian pada kondisi normal merepresentasikan situasi ideal ketika sistem beroperasi dalam lingkungan jaringan yang stabil.

Pada kondisi partisi parsial, koneksi antara *primary* dan salah satu *secondary* diputus menggunakan aturan *iptables*, sementara koneksi antara *primary* dan *secondary* lainnya tetap terjaga. Dengan demikian, mayoritas node (2 dari 3) tetap terpenuhi sehingga operasi tulis dengan konfigurasi *majority* masih dapat dijalankan. Skenario ini digunakan untuk mengevaluasi kemampuan sistem dalam mempertahankan ketersediaan layanan serta proses pemulihan setelah gangguan, yang diukur melalui *parameter request success rate, recovery time*, dan *rollback data*.

Parameter *rollback* data digunakan untuk mengidentifikasi kemungkinan terjadinya pembatalan atau pengembalian perubahan data akibat konflik replikasi atau proses *re-election* pada MongoDB *Replica Set*. Meskipun tidak selalu terjadi dalam setiap skenario pengujian, parameter ini tetap diukur untuk memastikan integritas data selama kondisi partisi jaringan.

Selain itu, *availability* tidak dianalisis sebagai variabel utama, karena telah direpresentasikan melalui parameter *request success rate* yang digunakan untuk mengukur keberhasilan layanan selama pengujian. Fokus utama penelitian tetap

diarahkan pada pengukuran *consistency* dan *partition tolerance* sebagai dua aspek yang menunjukkan *trade-off* secara langsung dalam sistem replikasi.

Pembagian skenario ini memungkinkan analisis empiris terhadap *trade-off* antara *consistency* dan *partition tolerance*, karena seluruh pengujian dilakukan pada topologi, spesifikasi perangkat, dan konfigurasi sistem yang sama, dengan perbedaan hanya pada kondisi jaringan yang diterapkan. Dengan pendekatan tersebut, perubahan nilai metrik dapat diinterpretasikan sebagai dampak langsung dari kondisi jaringan terhadap karakteristik replikasi sistem.

3.3.5 Sinkronisasi Waktu

Sinkronisasi waktu dilakukan untuk memastikan bahwa seluruh proses pengukuran metrik menggunakan acuan waktu yang *consistency*. Pada penelitian ini, semua *timestamp* dicatat dari sisi *client* menggunakan fungsi *time.time()* berbasis *UNIX epoch (UTC)* sehingga seluruh pengukuran menggunakan satu sumber waktu yang sama. Pendekatan ini menghilangkan potensi bias akibat perbedaan waktu lokal antar *server*, terutama ketika *primary* dan *secondary* memiliki *clock drift.*

Setiap operasi tulis dan baca pada pengujian *consistency dan partition tolerance* dipicu oleh *script* Python di *client*, sehingga selisih waktu (Δt) dihitung berdasarkan dua *timestamp* yang dicatat secara berurutan pada mesin yang sama. Dengan demikian, pengukuran *replication lag, latency strong, recovery time,* dan *stale reads* dapat dilakukan secara konsisten tanpa memerlukan protokol sinkronisasi eksternal seperti NTP.

Metode ini dipilih karena memberikan tingkat presisi tinggi, *reprodusibilitas*, serta memastikan hasil pengukuran murni dipengaruhi kondisi replikasi, bukan ketidaksamaan waktu antar *node*. Dari sisi *reprodusibilitas*, penggunaan satu sumber waktu terpusat memastikan bahwa prosedur pengukuran dapat diulang dengan mekanisme yang identik pada lingkungan eksperimen lain. Karena metode pencatatan waktu, parameter pengujian, dan skenario eksperimen didefinisikan secara eksplisit, peneliti lain yang menggunakan konfigurasi topologi dan beban kerja yang sebanding dapat mereplikasi proses pengukuran tanpa dipengaruhi perbedaan sinkronisasi antar *node*. Dengan demikian, variasi hasil yang diperoleh merefleksikan karakteristik sistem replikasi yang diuji, bukan perbedaan teknik pencatatan waktu, sehingga konsistensi metodologis dan validitas internal penelitian tetap terjaga.

3.4 Pengujian

Pengujian dilakukan pada dua kondisi utama, yaitu kondisi normal (tanpa gangguan jaringan) dan kondisi partisi parsial. Pada kondisi partisi parsial, koneksi antara *primary* dan salah satu *secondary* diputus menggunakan *iptables*, sementara *primary* tetap terhubung dengan *secondary* lainnya sehingga mayoritas (2 dari 3 *node*) tetap terpenuhi.

Setiap skenario diuji pada sepuluh variasi ukuran dataset JSON, yaitu 1.000, 2.000, 3.000, 4.000, 5.000, 6.000, 7.000, 8.000, 9.000, dan 10.000 dokumen. Setiap variasi dataset dijalankan sebanyak 30 kali untuk memperoleh hasil yang stabil secara statistik. Jumlah 30 dipilih karena secara umum telah memenuhi

pendekatan distribusi normal, sehingga analisis rata-rata dan deviasi standar dapat dilakukan secara representatif. Tabel skenario pengujian dapat dilihat pada tabel 3.4.

Tabel 3.4 Skenario Pengujian

Kondisi	**Aksi Sistem**	**Parameter yang Diukur**
Normal (Tanpa partisi)	Replikasi berjalan normal	*Replication Lag, Stale Reads, Latency Strong*
Partisi Parsial	Koneksi *primary* dan salah satu *secondary* diputus menggunakan *iptables*	*Request Success Rate, Recovery Time, Rollback Data*

Tabel 3.4 menggambarkan pembagian pengujian menjadi dua kondisi utama. Pada kondisi normal, seluruh *node* (*primary* dan *secondary*) saling terhubung sehingga proses replikasi berjalan tanpa gangguan. Kondisi ini digunakan untuk mengukur aspek *consistency*, yang mencakup *replication lag*, *stale reads*, dan *latency strong*. Skenario ini merepresentasikan kondisi ideal ketika tidak terjadi gangguan jaringan.

Pada kondisi partisi parsial, koneksi antara *primary* dan salah satu *secondary* diputus menggunakan *iptables*. *Secondary* yang terisolasi tidak dapat menerima replikasi, sedangkan *secondary* lainnya tetap terhubung dengan *primary* sehingga *quorum* tetap terpenuhi. Dengan demikian, operasi tulis menggunakan *writeConcern = "majority"* masih dapat dijalankan. Skenario ini digunakan untuk mengukur kemampuan sistem mempertahankan layanan dalam kondisi gangguan sebagian jaringan.

Pendekatan ini memungkinkan analisis *trade-off CAP* secara empiris karena setiap parameter diuji pada kondisi jaringan yang berbeda namun dengan dataset, topologi, dan konfigurasi yang sama.

3.4.1 Pengujian *Consistency*

Pengujian *consistency* bertujuan untuk mengukur karakteristik *consistency* data pada MongoDB *Replica Set* dalam kondisi jaringan normal. Parameter yang digunakan meliputi *replication lag, stale reads,* dan *latency strong*.

Replication lag diukur berdasarkan selisih waktu antara operasi tulis pada *primary* dan waktu data tersebut tersedia pada *secondary*. Pengukuran dilakukan dengan mencatat *timestamp* saat operasi *insert* dijalankan pada *primary* dan membandingkannya dengan waktu kemunculan dokumen pada *secondary*.

Stale reads diukur dengan menghitung persentase pembacaan data lama pada *secondary* ketika replikasi belum sepenuhnya tersinkronisasi. Untuk keperluan ini digunakan konfigurasi *readPreference = "secondary*" agar pembacaan dilakukan langsung dari *secondary*.

Latency strong diukur sebagai waktu yang dibutuhkan sejak operasi tulis dijalankan dengan *writeConcern = "majority"* hingga sistem memberikan konfirmasi keberhasilan tulis. Parameter ini merepresentasikan biaya latensi untuk memperoleh jaminan konsistensi mayoritas.

3.4.2 Pengujian *Partition Tolerance*

Pengujian *partition tolerance* dilakukan dengan mensimulasikan gangguan jaringan antar node menggunakan *iptables* secara terkontrol. Pemutusan koneksi

dilakukan hanya antara *primary* dan salah satu *secondary*, sehingga satu *secondary* tetap terhubung dan *quorum* tetap terpenuhi. Parameter yang diamati meliputi *request success rate, recovery time*, dan *rollback data*. *Request success rate* menunjukkan persentase perintah yang berhasil dijalankan selama terjadi partisi parsial. *Recovery time* dihitung berdasarkan selisih waktu sejak koneksi dipulihkan hingga seluruh node kembali dalam kondisi sinkron. *Rollback data* digunakan untuk mengukur kemungkinan terjadinya pembatalan atau pengembalian perubahan data akibat konflik replikasi.

Peraturan pemutusan jaringan:

```
sudo iptables -A INPUT  -p tcp --dport 27017 -s <IP_NODE> -j REJECT
sudo iptables -A OUTPUT -p tcp --dport 27017 -d <IP_NODE> -j REJECT
```

Pemulihan jaringan:

```
sudo iptables -D INPUT  -p tcp --dport 27017 -s <IP_NODE> -j REJECT
sudo iptables -D OUTPUT -p tcp --dport 27017 -d <IP_NODE> -j REJECT
```

Selama partisi, *primary* tetap menerima operasi tulis, sementara *secondary* tidak dapat melakukan sinkronisasi. Setelah koneksi dipulihkan, *secondary* melakukan proses catch-up atau *rollback* tergantung status oplog, sehingga nilai ketiga parameter dapat diamati secara akurat.

3.5 Analisis Hasil

Analisis hasil dilakukan dengan mengolah data dari pengujian *consistency* dan *partition tolerance* untuk membandingkan kinerja sistem pada berbagai skenario. Data disajikan dalam bentuk tabel dan grafik agar lebih mudah dipahami dan dianalisis. Perbandingan difokuskan pada pengaruh jumlah data dan kondisi jaringan terhadap parameter yang diukur. Dari analisis ini dapat dilihat pola hubungan antara beban sistem dan performa *replica set* MongoDB.