BAB II
LANDASAN TEORI

2.1 Teori Tentang Obyek Penelitian

2.1.1 *Database*

Database adalah kumpulan data yang sistematis dimana *database* ini dapat memberikan layanan berupa peyimpanan hingga manipulasi data yang bertujuan untuk mempermudah proses manajemen data. Proses manajemen data ini dibantu oleh DBMS (*Database Management System*) yang merupakan satu set atau kumpulan dari program yang memungkinkan *user* untuk melakukan proses akses hingga *reporting* dari data yang tersimpan didalam *database* itu sendiri. [7]

2.1.2 *Connection Pooling*

Connection pooling merupakan sebuah *middleware* yang berada diantara *layer* aplikasi dan juga *layer database*, *connection pooling* berperan sebagai *middleware* yang melakukan manajemen koneksi dari aplikasi kedalam *database* [8]. *Connection pooling* menyediakan *pool* (penampungan) yang dapat menampung koneksi sebelum masuk kedalam *database* dan juga dapat melakukan rekayasa koneksi antara *database* dengan aplikasi [9].

2.1.3 *Load balancing*

Load balancing merupakan *middleware* yang bertugas untuk membagikan atau mendistribusikan beban transaksi ke dalam beberapa server yang terkoneksi berdasarkan dengan spesifikasi atau kemampuan dari setiap server tersebut. Tujuan dari *load balancing* ini adalah untuk menghindari *down server* jika terjadi penumpukan transasksi pada 1 server saja [10].

2.1.4 *Benchmarking*

Benchmarking adalah suatu upaya untuk mengukur atau menentukan standar yang dilakukan dengan membandingkan hal sejenis dan dapat digunakan sebagai tolak ukur. Tujuan dari *benchmarking* sendiri adalah untuk mendapatkan nilai dasar dan bisa dijadikan sebagai patokan untuk perkembangan kedepannya [11].

2.1.5 *Latency*

Latency merupakan jarak waktu yang berada diantara respon *output* yang didapatkan dari *input* yang diberikan dengan kata lain *Latency* menggambarkan waktu yang dibutuhkan untuk mendapatkan respon *output* berdasarkan dengan *input* yang diberikan dan juga menggambarkan seberapa cepat proses yang berjalan dalam aplikasi dari *input* yang diberikan oleh *user*, hingga *user* mendapatkan respon yang diharapkan dari proses aplikasi yang digunakan [12].

2.1.6 *Load Test*

Load Test merupakan salah satu bentuk dari *performance test* yang merupakan *test* pada sistem menggunakan beberapa angka yang sudah ditentukan sebelumnya yang dimana angkat tersebut diambil untuk menggambarkan performa dari sistem pada satuan waktu tertentu secara bersamaan. *Load Test* dilakukan dengan asumsi beberapa skenario jumlah *user* tertentu yang mungkin dapat terjadi dalam sistem dan memperlihatkan bagaimana sistem menangani jumlah *user* tersebut pada waktu yang bersamaan [13].

2.1.7 *Transaction per Second* (TPS)

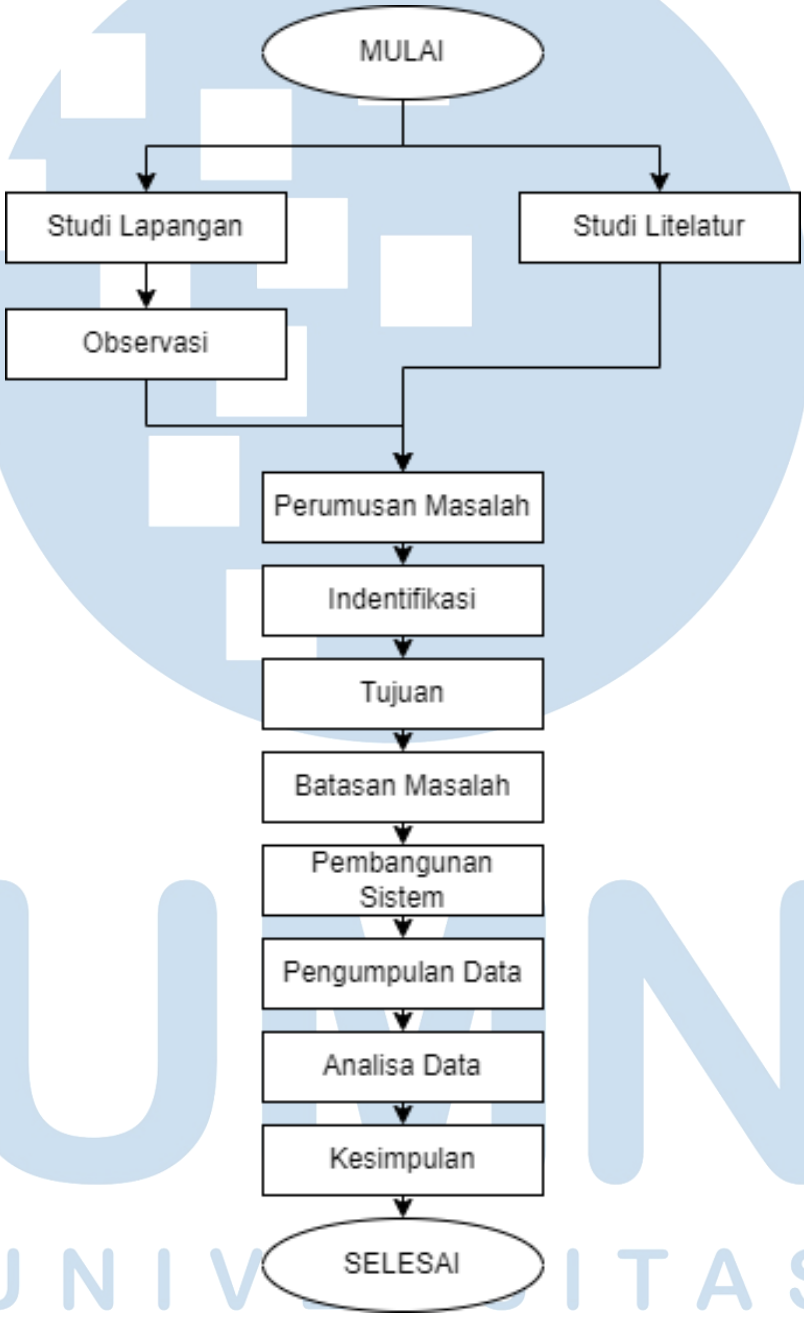
Transaction per Second (TPS) merupakan angka yang menggambarkan kemampuan suatu sistem untuk memproses beberapa transaksi dalam satu detik yang sama. TPS biasa digunakan untuk mengukur atau memperlihatkan bagaimana performa suatu sistem [14].

2.1.8 *Asynchronous Replication*

Replikasi Asinkronus merupakan salah satu dari jenis cara melakukan replikasi pada *database*. Replikasi dari *database* sendiri merupakan sebuah usaha yang dilakukan untuk mencapai *High Avaibility* (HA) yang merupakan kemampuan dari *database* untuk terus aktif pada kondisi yang tak terduga, replikasi merupakan upaya mencapai HA dengan membuat *backup* data yang sama dengan data asli yang bertempat di server atau tempat lain selain *database* utama. Replikasi asinkronus bergerak setalah transaksi selesai dilakukan *commit* pada *database* utama, setelah *database* utama melakukan *commit* maka data tersebut juga akan diduplikasi ke *database backup*. Replikasi *Asynchronous* dipilih karena memiliki performa yang lebih baik dimana transaksi pada master tidak perlu menunggu transaksi pada slave untuk selesai terlebih dahulu dan juga cocok untuk *environment* internal Perusahaan X [15].

2.2 Diagram Alir Penelitian

Berikut merupakan gambaran *Framework* penelitian ini:



Gambar 2. 1 Diagram Alir Penelitian

Gambar 2.1 merupakan gambaran Diagram Alir Penelitian yang akan digunakan sebagai pedoman dalam penelitian ini.

2.3 *Tools* yang digunakan

2.3.1. Redhat Enterprise Linux 8

Redhat Enterprise Linux 8 (RHEL 8) merupakan sistem operasi dengan basis linux yang dikembangkan oleh perusahaan Redhat, RHEL 8 ini rilis pada tahun 2019 lalu yang menggunakan kernel linux versi 4.18 [16]. Pada penelitian ini RHEL 8 dijadikan basis sistem operasi untuk membangun environment *database* nantinya. Alasan penggunaan RHEL 8 ini karena:

1. Redhat merupakan *official partner* dari *vendor* yang melakukan manajemen *database* PostgreSQL di Perusahaan X.
2. RHEL 8 memiliki *support* 24 jam dari Redhat.
3. RHEL 8 sudah teruji dan banyak digunakan dalam lingkungan produksi sistem dibanyak perusahaan.
4. Merupakan sistem operasi berbasis *open-source* dan memiliki lingkungan pengenambangan yang *flexible* [16].

2.3.2. PostgreSQL *Database*

PostgreSQL merupakan sebuah sistem manajemen basis data relasional yang populer dengan sumber terbuka atau *open source*. Sistem ini mengadopsi model relasional (RDBMS), di mana data disimpan dalam bentuk tabel dan kolom yang terstruktur dan terhubung melalui kunci asing. PostgreSQL memiliki banyak fitur yang mirip dengan sistem manajemen basis data komersial seperti Oracle atau SQL Server, seperti fitur pemulihan bencana (DRC), replikasi, dan tuning performa [17].

2.3.3. PGBOUNCER

PGOUNCER merupakan perangkat *open-source* yang memungkinkan *database* PostgreSQL memiliki fitur tambahan yaitu *connection pooling*. PGBOUNCER sendiri merupakan *connection pooler* yang hanya khusus digunakan untuk *database* berbasis PostgreSQL [18]. Pada penelitian ini PGBOUNCER akan

digunakan sebagai *tools connection pooler* pada *database* PostgreSQL. PGBOUNCER akan diinstalasikan pada server terpisah dengan server *database* PostgreSQL untuk memudahkan pemantauan pengguaan resource server pada penelitian ini.

2.3.4. Check Point Capsule VPN

Check Point Capsule VPN merupakan *software* VPN yang digunakan untuk mengakses internet internal milik peruhaan sehingga server yang ada di dalam *data center* milik perusahaan dapat diakses tanpa harus menggunakan koneksi internal perusahaan.

2.3.5. MobaXterm

MobaXterm merupakan *software* yang dapat digunakan untuk SSH *client* yang gratis. MobaXterm sendiri dibuat untuk memudahkan *programmer* hingga *server administration* untuk melakukan *remote* SSH dengan mudah dengan berbagai tipe koneksi yang dapat digunakan, multi-execution dan banyak fitur lainnya [19].

2.3.6. PGPOOL-II

PGPOOL-II merupakan *tools* atau perangakat lunak yang memungkinkan *database* untuk memiliki fitur *connection pooling* dan *load balancing*. PGPOOL-II digunakan hanya untuk *database* berbasis PostgreSQL [20]. Pada penelitian ini akan dibandingkan kombinasi *connection pooling* dan *load balancing* dengan menggunakan *connection pooling* dari PGPOOL-II dan *connection pooling* dari PGBOUNCER sedangakan *load balancing* akan tetap menggunakan *load balancer* dari PGPOOL-II.

2.3.7. PGBENCH

PGBENCH menrupakan *tools* tambahan atau fitur yang terdapat di dalam *database* PostgreSQL, PGBENCH dapat digunakan untuk melakukan pengujian kinerja dari *database* PotgreSQL dimana metode pengujian ini mirip dengan metode *performance test* yaitu

load test dengan menggunakan beberapa skenario tertentu. Dalam penelitian ini PGBENCH akan digunakan sebagai alat untuk menghimpun data TPS dan *latency* dari penggunaan PGPOOL-II dan juga PGBOUNCER pada *database* PostgreSQL. TPS dan *latency* ini akan digunakan untuk membandingkan skenario mana yang paling baik dan dijadikan *best practice* untuk Perusahaan X dengan asumsi TPS tertinggi dan juga *latency* terendah merupakan skenario terbaik [20].

2.4 Penelitian Terdahulu

Tabel 2. 1 Penelitian Terdahulu

No	Judul Penelitian	Infromasi Tambahan	Hasil Penelitian
1	*High Availibility and Load balancing For PostgreSQLDatabases: Designing and Implementing*	Tahun: 2016 Jurnal: International Journal of Database Management Systems Volume: 8 No: 6 Penulis: Pablo Bárbaro Martinez Pedroso	Dalam penelitian ini dibangun dan diimplementasikan solusi untuk menyediakan ketersediaan (HA) dan *Load Balancer* yang tinggi untuk *database* PostgreSQL. Oleh karena itu, menggunakan layanan *database* PostgreSQL yang lebih efisien. Dengan memiliki beberapa replikasi *database* dengan salinan lengkap data tidak hanya memungkinkan mekanisme *failover*, tetapi juga penyeimbangan beban di antara server. Setiap server memiliki salinan data yang lengkap dan konsisten, masing-masing dapat memproses semua jenis transaksi yang masuk ke dalam *database*. Situasi ini menghindari kerusakan layanan karena penyebab kelebihan beban. Penggunakan lebih dari satu *middleware* (satu aktif, setidaknya satu *stand-by*) menghilangkan titik kegagalan tunggal, sehingga ketersediaan tinggi dapat dicapai pada tingkat *middleware*. Arsitektur *database* terdistribusi bersifat tidak bersamaan, banyak server dapat ditambahkan atau dihapus dalam beberapa langkah konfigurasi, karena pendekatan yang disajikan fleksibel dan dapat diskalakan. Semua alat yang dipilih adalah open source dan dikembangkan untuk Linux, sehingga solusi dari investigasi ini dimaksudkan hanya untuk Sistem Operasi berbasis Linux [5].

No	Judul Penelitian	Infromasi Tambahan	Hasil Penelitian
2	*Performance Evaluation of MongoDB and PostgreSQL for spatio-temporal data*	Tahun: 2020 Workshop Proceedings of the EDBT/ICDT 2019 Joint Conferenceon Volume: 2322 Penulis: Antonios Makris, Konstantinos Tserpes, Giannis Spiliopoulos dan Dimosthenis Anagnostopoulos	Penelitian ini membahas mengenai permasalahan yang terkait dengan pengolahan data spasio-temporal dan mencari sistem penyimpanan data yang paling efisien dalam hal waktu respons. Pada penelitian ini *database* PotgreSQL menggunakan *tools* PGPOOL-II sebagai *tools* untuk *connection pooling*nya. Penelitian ini membandingkan MongoDB dan PostgreSQL pada skenario bisnis yang nyata serta infrastruktur yang mendasarinya, dan hasilnya menunjukkan bahwa PostgreSQL lebih unggul dibandingkan dengan MongoDB [21].
3	*Teknik Connection pooling Untuk Meningkatkan Performa Aplikasi Dengan Basis Data Jarak Jauh*	Tahun: 2019 Prosiding SENSITIf 2019 Seminar Nasional Sistem Informasi dan Teknik Informatika Penulis: Tony Wijaya	Penelitian ini membahas mengenai pemrograman dengan basis data (*database*) jarak jauh, dengan menggunakan teknik khusus untuk berkomunikasi dengan server yang berada di *cloud*. Penelitian ini bertujuan untuk memaparkan teknik *connection pooling* pada pemrograman demi meningkatkan performa aplikasi dengan basis data yang sangat jauh. Dalam perancangan, digunakan metode *Agile* dengan pendekatan *Extreme Programming* yang lebih mengedepankan tercapainya fitur yang akan dibangun. Berdasarkan hasil dan pembahasan, dapat disimpulkan bahwa implementasi teknik *connection pooling* pada aplikasi desktop dengan basis data jarak jauh berhasil meningkatkan performa aplikasi secara signifikan. Hal ini memungkinkan aplikasi dapat berfungsi sebagaimana mestinya seperti pada saat basis data masih berada pada jaringan *Local Area Network* (LAN). Selain itu, hasil implementasi juga menghilangkan error pada aplikasi akibat kegagalan koneksi yang sangat intensif [22].

No	Judul Penelitian	Infromasi Tambahan	Hasil Penelitian
4	*Load balancing in cloud computing – A hierarchical taxonomical classification*	Tahun: 2019 Jurnal: Journal of Cloud Computing Volume: 8 No: 1 Penulis: Shahbaz Afzal dan G. Kavitha	Penelitian ini membahas sebuah studi perbandingan mengenai pendekatan *load balancing* pada artikel-artikel yang ditinjau. Masalah *load unbalancing* dalam *cloud computing* dibahas secara rinci beserta faktor-faktor pendorong yang menyebabkannya. Selain itu, model *load balancing* yang disederhanakan juga dijelaskan, termasuk aktivitas yang terlibat dalam prosesnya. Metodologi penelitian yang digunakan dalam penelitian ini adalah *Framework Generic Konstruktif* (CGF) yang diperkuat dengan metodologi *Systematic Literature Review* (SLR). Pertanyaan-pertanyaan terkait masalah juga dirumuskan dan dijawab. Data untuk penelitian ini dikumpulkan dari lima basis data yang berbeda, dan proses pencarian data dibantu dengan berbagai alat dan filter opsional. Dalam studi ini, diusulkan klasifikasi *multilevel* yang dibagi berdasarkan lima kriteria. Dalam klasifikasi ini, terdapat 35 artikel yang dibagi menjadi dua kategori umum: 10 proaktif dan 25 reaktif. Selain itu, penjadwalan tugas juga diberikan banyak perhatian baik dalam pendekatan proaktif maupun reaktif, yang masing-masing memberikan kontribusi 45% dan 51,85%. [23].

No	Judul Penelitian	Infromasi Tambahan	Hasil Penelitian
5	Analisis Metode *Load balancing* Dalam Meningkatkan Kinerja *Website E-LEARNING*	Tahun: 2020 Jurnal: Jurnal Teknoinfo Volume: 14 No: 1 Penulis: Sampurna Dadi Riskiono dan Donaya Pasha	Penelitian ini berdasar pada maslah pendidikan saat ini yang berkembang pesat, terlihat dari penggunaan teknologi informasi yang semakin meningkat. Pemanfaatan teknologi informasi dalam pembelajaran membuat proses belajar mengajar menjadi lebih interaktif dan menarik. Awalnya, teknologi informasi digunakan hanya untuk menyampaikan materi melalui presentasi Power Point, Adobe Flash, atau aplikasi serupa. Namun, dengan adanya internet, penggunaan aplikasi yang berbasis online semakin banyak digunakan, seperti e-learning. Namun, masalah terjadi saat jumlah pengguna yang mengakses layanan e-learning semakin banyak dan server tidak dapat mengatasinya. Oleh karena itu, perlu ada sistem server yang dapat menangani banyak permintaan layanan untuk meningkatkan skalabilitas dari server e-learning. Salah satu solusinya adalah dengan menerapkan *load balancing*. Penelitian ini mengevaluasi penggunaan server tunggal dan server jamak. Hasil dari penelitian ini menggambarkan bahwa implementasi dari fitur *load balancer* memiliki nilai *response time* 36,4 ms lebih kecil jika disandingkan dengan server tunggal yang memiliki *response time* sebesar 51,1 ms dengan uji koneksi di 500/10 detik. Hasil dari pengujian yang dilakukan menunjukan penerapan fitur *load balancer* lebih baik dari pada server tunggal [24].

No	Judul Penelitian	Infromasi Tambahan	Hasil Penelitian
6	*The Design of Sentiment Analysis Application using Top-Down Development Approach*	Tahun: 2021 2021 6th International Conference on New Media Studies (CONMEDIA) Penulis: Aria Eka Putra, Suryasari	Penelitian ini membahas mengenai perbandingan performa dari *database* NoSQL. Perbandingan perform ini dilakukan pada database MongoDB, CouchDB dan RavenDB, dengan menggunakan data sentimen yang berasal dari twitter dalam bentuk JSON dengan operasi yang dilakukan adalah Getmore dan *Insert*. Berdasarkan data *response time* dalam satuan *millisecond* didapatkan bahwa MongoDB memiliki kinerja kurang baik jika dibandingkan dengan RavenDB dan CouchDB. Nilai response yang yang didapatkan untuk *database* MongoDB mancapai 16.8000 *millisecond* dengan jumlah data 151.200 data JSON. Hasil evaluasi ini tidak menggambarkan bahwa kinerja dari MongoDB lebih baik dibanding dengan *database* NoSQL lainnya karena *environment* evaluasi yang digunakan tidak sama dengan *environment* yang digunakan untuk *testing* di RavenDB dan CouchDB dimana *resource* yang digunakan berbeda.[25]

No	Judul Penelitian	Infromasi Tambahan	Hasil Penelitian
7	Perbandingan Hasil Penggunaan Metode *Decision Tree* dan *Random Tree* Pada Data Training Aplikasi Pencarian Tukang	Tahun: 2019 Jurnal: ULTIMA InfoSys Volume: 10 No: 2	Decision tree dan random tree merupakan metode yang umum digunakan dalam pembuatan pohon keputusan. Decision tree menggunakan parameter-parameter seperti max depth (kedalaman maksimum), confidence (kepercayaan), dan minimal gain (keuntungan minimal) untuk membangun struktur pohon yang berdasarkan pemilihan fitur terbaik. Sebaliknya, random tree tidak memiliki parameter-parameter yang spesifik. Melalui analisis yang mendalam, kami menemukan bahwa decision tree memiliki keunggulan dalam aplikasi pencarian tukang. Dalam pengujian kami, decision tree mencapai tingkat akurasi sebesar 67% dengan menggunakan parameter-parameter yang telah ditentukan. Hal ini menunjukkan bahwa decision tree mampu memberikan hasil yang dapat diandalkan dalam memilih tukang yang sesuai dengan permintaan pelanggan dan mempertimbangkan posisi geografis tukang tersebut.[26]

Penelitian ini berdasar pada 5 penelitian sebelumnya, yang dimana kelima penelitian tersebut membahas fitur *connection pooling* dan *load balancer* secara terpisah, fitur *load balancer* pada penelitian 4 dan 5 juga diterapkan pada server dari *service* yang dijalankan tidak spesifik langsung ke dalam *service* yang diinginkan. Penelitian ke 6 memberikan gambaran bagaimana cara untuk melakukan evaluasi penggunaan database dengan membandingkan produk database yang berbeda kemudian untuk penelitian ke 7 memberikan gambaran bagaimana untuk mengevaluasi sistem machine learning dengan menggunakan metode yang sejenis tetapi dapat mendapatkan metode mana yang lebih baik.

Penelitian ini ingin menarik benang merah dari 7 penelitian diatas dengan melakukan tes performa dari kombinasi ke-dua fitur tersebut dan pemasangan *load balancer* juga spesifik langsung ke dalam *database* yang ingin digunakan bukan pada server tempat *database* tersebut di *install*.