BAB I

PENDAHULUAN

1.1 Latar Belakang

Di era transformasi digital yang terus berkembang pesat, kemampuan organisasi dalam mengelola data skala besar menjadi fondasi penting dalam mendukung operasional dan pengambilan keputusan yang cepat dan tepat [1]. Hampir seluruh aktivitas bisnis kini bergantung pada sistem informasi yang dibangun di atas *database* yang efisien, stabil, dan responsif terhadap beban kerja tinggi [2]. *Database* menjadi komponen fundamental dalam sistem informasi masa kini [3]. Dalam konteks tersebut, PostgreSQL menjadi salah satu sistem manajemen *database* relasional (RDBMS) *open-source* yang populer karena memiliki dukungan fitur lengkap, termasuk *indexing*, *partitioning*, parallel *query execution*, serta fleksibilitas untuk berbagai jenis data [4], [5].

Namun, seiring dengan meningkatnya volume data dan kompleksitas transaksi, tantangan performa menjadi tidak terhindarkan. Permasalahan seperti lambatnya eksekusi *query*, peningkatan latensi sistem, dan beban pemrosesan yang tinggi sering kali muncul ketika *database* tidak dioptimalkan dengan baik [6]. Dalam skenario seperti ini, sistem *database* harus melakukan pemindaian tabel secara menyeluruh untuk menjalankan *query*, yang dikenal sebagai sequential scan, dan ini sangat tidak efisien untuk dataset berskala besar [7].

Dua teknik utama yang terbukti efektif dalam meningkatkan kinerja PostgreSQL adalah *indexing* dan *partitioning* [4]. *Indexing* berfungsi mempercepat pencarian data melalui struktur data tambahan pada kolom tertentu, sehingga *query* tidak perlu menelusuri seluruh tabel. Dalam studi yang dilakukan oleh Martins, penggunaan *indexing* dapat meningkatkan performa PostgreSQL hingga 91% pada skenario benchmark TPC-H [7]. Ini menunjukkan bahwa strategi *indexing* memberikan kontribusi signifikan dalam mengurangi waktu eksekusi *query* pada volume data yang besar.

Di sisi lain, *partitioning* adalah teknik untuk membagi tabel besar menjadi beberapa partisi berdasarkan nilai tertentu seperti *range*, *list*, atau hash, sehingga sistem hanya membaca subset data yang relevan saat menjalankan *query* [4].

Namun, efektivitas *tuning* tidak hanya bergantung pada penerapan *indexing* dan *partitioning* saja, tetapi juga pada pemilihan jenis struktur *index* yang sesuai dengan pola akses data. Misalnya, B-Tree *index* terbukti sangat baik untuk operasi *SELECT* dan *UPDATE*, GIN *index* lebih optimal untuk pencarian data bertipe JSONB, sementara BRIN *index* cocok untuk data dalam jumlah besar yang memiliki urutan alami. Studi tersebut juga menyarankan pemilihan *index* disesuaikan dengan jenis operasi dan volume data untuk mencapai efisiensi maksimal [8].

Studi-studi di Indonesia juga mulai mengadopsi teknik serupa, seperti penelitian oleh Boymau pada sistem STARS Universitas Kristen Satya Wacana. Hasil penelitian menunjukkan bahwa vertical dan *list partitioning* mampu meningkatkan kecepatan operasi *SELECT* , *UPDATE*, dan *DELETE* secara signifikan. Namun, performa *INSERT* sedikit menurun pada tabel yang menggunakan partisi, sehingga pemilihan skema partisi harus mempertimbangkan jenis beban kerja sistem [9].

Terlepas dari bukti ilmiah yang kuat tentang manfaat *indexing* dan *partitioning*, kenyataannya masih banyak perusahaan yang belum memanfaatkannya secara optimal. Faktor-faktor seperti keterbatasan sumber daya manusia, kurangnya dokumentasi performa, dan belum adanya standar praktik *tuning* yang menyeluruh menjadi kendala dalam implementasinya [10]. PT ABC merupakan salah satu contoh perusahaan yang menghadapi permasalahan serupa, di mana sistem PostgreSQL yang digunakan mengalami perlambatan performa yang berdampak langsung pada aplikasi operasional harian.

PT ABC merupakan perusahaan kontraktor yang menangani transaksi proyek berskala besar setiap harinya. Oleh karena itu, penelitian ini akan menerapkan strategi performance *tuning* menggunakan *indexing* dan *partitioning* pada *database* PostgreSQL milik PT ABC. Pendekatan ini diharapkan tidak hanya mempercepat waktu respon sistem, tetapi juga dapat menjadi pedoman praktis dalam implementasi *tuning* di perusahaan lain yang menghadapi tantangan serupa. Pada

skema LIST partitioning, penambahan data dengan nilai kategori baru memerlukan pembuatan partisi tambahan secara eksplisit. Hal ini menjadi trade-off dari LIST partitioning, namun tetap relevan digunakan pada kolom dengan jumlah kategori terbatas dan stabil seperti *isactive*, *processing*, atau *docstatus*. Dalam konteks PT ABC, kategori tersebut jarang berubah sehingga LIST partitioning masih efisien dan mudah dikelola. Penelitian ini membatasi ruang lingkup pada teknik indexing dan partitioning untuk menjaga fokus dan menghindari bias dari parameter tuning lain seperti *shared_buffers*, *work_mem*, atau *parallel query*. Dengan demikian, peningkatan performa yang dihasilkan dapat dikaitkan secara langsung dengan efek struktur data dan skema partisi.

1.2 Rumusan Masalah

1. Bagaimana penerapan teknik *indexing* dan *partitioning* dapat mempengaruhi peningkatan performa sistem *database* PostgreSQL pada PT ABC?
2. Bagaimana mekanisme *partitioning* yang sesuai dapat meningkatkan efisiensi memanggil data (*SELECT*) dan mendukung kinerja *query* pada PostgreSQL di PT ABC?
3. Seberapa besar peningkatan performa yang dapat dicapai setelah dilakukan *tuning* menggunakan kombinasi *indexing* dan *partitioning* pada PostgreSQL di PT ABC?

1.3 Batasan Masalah

1. Penelitian ini hanya menggunakan PostgreSQL sebagai sistem manajemen *database* (DBMS) yang berbasis *Relational Database Management System* (RDBMS).
2. Penelitian hanya difokuskan pada penerapan dua teknik performance *tuning*, yaitu *indexing* dan *partitioning*.
3. Lingkungan pengujian terbatas pada sistem *database* PostgreSQL yang digunakan oleh PT ABC.
4. Jenis *index* yang dianalisis dibatasi pada *index* bawaan PostgreSQL, salah satunya *index* yang digunakan adalah B-Tree.

5. Skema *partitioning* yang digunakan difokuskan pada teknik declarative *partitioning* yang didukung sejak PostgreSQL versi 10 ke atas, seperti *range*, *list*, dan hash *partitioning*, sesuai dengan fitur bawaan PostgreSQL tanpa penggunaan ekstensi tambahan.

1.4 Tujuan dan Manfaat Penelitian

1.4.1 Tujuan Penelitian

1. Menerapkan teknik *indexing* dan *partitioning* pada *database* PostgreSQL milik PT ABC untuk meningkatkan performa sistem dalam menjalankan *query* terhadap dataset berskala besar.
2. Mengidentifikasi jenis partisi yang paling sesuai dengan struktur data dan pola *query* yang umum terjadi pada sistem informasi PT ABC.
3. Mengukur dampak dari penerapan *indexing* dan *partitioning* terhadap performa sistem, khususnya dalam hal peningkatan kecepatan eksekusi *query* dan efisiensi akses data.

1.4.2 Manfaat Penelitian

1. Penelitian ini dapat memperkaya kajian ilmiah dalam bidang sistem *database*, khususnya terkait strategi performance *tuning* menggunakan teknik *indexing* dan *partitioning* pada PostgreSQL.
2. Memberikan panduan teknis dalam menerapkan *indexing* dan *partitioning* untuk meningkatkan performa sistem *database*.
3. Memberikan solusi nyata untuk perusahaan dalam mengatasi masalah *database* lambat ketika mengambil data (*SELECT*).

1.5 Sistematika Penulisan

Berikut adalah sistematika penulisan yang menjelaskan isi dari setiap bab dalam penelitian ini.

BAB I PENDAHULUAN

Bab ini berisi latar belakang, rumusan masalah, batasan masalah, tujuan penelitian, manfaat penelitian, serta sistematika penulisan. Bagian pendahuluan

menjelaskan alasan mengapa topik penelitian ini penting untuk diangkat, sekaligus memberikan gambaran umum mengenai arah penelitian yang akan dilakukan.

BAB II LANDASAN TEORI

Bab ini memuat landasan teori dan referensi ilmiah yang mendukung penelitian. Dalam bab ini dijelaskan konsep dasar sistem manajemen *database*, khususnya PostgreSQL, serta pembahasan mengenai teknik *indexing* dan *partitioning* sebagai strategi optimasi performa. Selain itu, bab ini juga memuat studi literatur terdahulu yang relevan, sebagai bentuk *state of the art* penelitian, serta disusun kerangka pemikiran yang menjadi acuan dalam pelaksanaan dan analisis penelitian.

BAB III METODOLOGI PENELITIAN

Bab ini membahas pendekatan yang digunakan dalam melaksanakan penelitian ini. Bab ini mencakup jenis penelitian, objek dan lokasi studi kasus (PT ABC), metode pengumpulan data, serta tahapan implementasi strategi performance *tuning* menggunakan *indexing* dan *partitioning* pada PostgreSQL. Selain itu, dijelaskan pula metode evaluasi performa *database* sebelum dan sesudah *tuning*, serta perangkat lunak, alat ukur, dan parameter yang digunakan dalam proses pengujian.

BAB IV ANALISIS DAN HASIL PENELITIAN

Bab ini menyajikan hasil implementasi strategi *tuning* yang telah dilakukan dan analisis performa sistem berdasarkan data empiris. Bab ini menguraikan kondisi awal *database* PT ABC sebelum dilakukan optimasi, hasil penerapan *indexing* dan *partitioning*, serta analisis perbandingan performa sistem. Pembahasan dilakukan dengan mengaitkan hasil yang diperoleh dengan teori dan penelitian terdahulu untuk memperkuat validitas temuan.

BAB V SIMPULAN DAN SARAN

Bab ini berisi kesimpulan dari hasil penelitian yang diperoleh berdasarkan tujuan dan rumusan masalah. Selain itu, bab ini juga memuat saran-saran yang dapat dijadikan pertimbangan bagi pengembangan sistem *database* di masa

mendatang, baik oleh pihak PT ABC maupun oleh penelitian selanjutnya yang tertarik pada topik serupa.