BAB V

SIMPULAN DAN SARAN

5.1 Kesimpulan

Berdasarkan hasil pengujian performa yang telah dilakukan pada sepuluh tabel utama sistem database PostgreSQL PT ABC, dapat disimpulkan bahwa penerapan teknik *indexing* dan *partitioning* memberikan peningkatan performa yang signifikan, khususnya pada *query* bertipe *SELECT* dengan beban kerja dominan (*SELECT-heavy workload*). Pengujian dilakukan menggunakan perintah *EXPLAIN ANALYZE* dengan lima kali eksekusi (*loops* = 5) pada setiap skenario sebelum dan sesudah optimasi, sehingga nilai *execution time* yang diperoleh bersifat stabil dan representatif.

Menjawab rumusan masalah pertama, yaitu bagaimana perbedaan performa query sebelum dan sesudah penerapan *indexing* dan *partitioning*, hasil penelitian menunjukkan bahwa seluruh tabel mengalami penurunan waktu eksekusi, namun dengan tingkat peningkatan yang berbeda-beda. Tabel dengan volume data besar dan pola akses yang jelas menunjukkan peningkatan paling signifikan. Sebagai contoh, tabel bpartner mengalami penurunan waktu eksekusi dari 154.615 ms menjadi 0.309 ms (penurunan ±99,8%), tabel m_reqorder dari 1025.95 ms menjadi 1170 ms (penurunan ±99,9%), serta tabel m_req dari 1141.28 ms menjadi 8.411 ms (penurunan ±99,3%). Sementara itu, tabel dengan ukuran relatif kecil seperti m_area dan m_product tetap menunjukkan peningkatan signifikan, meskipun nilai absolut waktu eksekusinya sejak awal sudah rendah.

Menjawab rumusan masalah kedua, penelitian ini menemukan bahwa tidak semua tabel mengalami peningkatan performa hingga di atas 90%, namun sebagian besar tabel kritikal dengan beban akses tinggi mencapai atau bahkan melampaui angka tersebut. Tabel aduser, c_orderline, m_req, m_reqline, dan m_reqorder menunjukkan penurunan waktu eksekusi lebih dari 90%, sedangkan tabel c_project dan c_order menunjukkan peningkatan yang lebih moderat, masing-masing sekitar 82% dan 64%. Hal ini menunjukkan bahwa

efektivitas optimasi sangat dipengaruhi oleh karakteristik data, ukuran tabel, serta selektivitas kondisi *query* yang digunakan.

Menjawab rumusan masalah ketiga, penelitian ini membuktikan bahwa pemilihan jenis partisi harus disesuaikan dengan karakteristik tabel dan pola *query*. *Range partitioning* terbukti paling efektif untuk tabel berbasis waktu seperti c_order, c_orderline, dan c_project, karena mampu membatasi ruang pencarian data secara signifikan. *List partitioning* paling sesuai untuk tabel dengan kategori terbatas seperti bpartner dan m_req, sedangkan hash partitioning memberikan performa paling stabil pada tabel dengan distribusi data acak dan pencarian berbasis ID seperti aduser dan m_reqline.

Secara keseluruhan, penelitian ini menyimpulkan bahwa kombinasi *indexing* dan *partitioning* terbukti efektif, namun tidak bersifat seragam untuk semua tabel. Peningkatan performa tertinggi dicapai pada tabel berukuran besar dengan pola akses yang konsisten, sedangkan tabel dengan ukuran kecil atau selektivitas rendah tetap mendapatkan peningkatan, namun dalam skala yang lebih terbatas. Dengan demikian, tujuan penelitian telah tercapai, yaitu mengukur dampak nyata optimasi struktur *database*, mengidentifikasi jenis partisi yang paling sesuai, serta memberikan bukti empiris bahwa teknik indexing dan partitioning relevan dan layak diterapkan pada sistem operasional PT ABC.

Selain itu, selama proses penelitian ditemukan bahwa penerapan optimasi pada lingkungan operasional nyata memiliki tantangan tersendiri, khususnya terkait keterbatasan akses terhadap sistem produksi dan risiko gangguan layanan. Oleh karena itu, seluruh pengujian dilakukan pada *environment* replikasi dengan data *dummy* yang merepresentasikan karakteristik data asli, sehingga hasil penelitian tetap valid namun aman untuk sistem operasional.

5.2 Saran

Berdasarkan hasil penelitian dan keterbatasan yang ditemui, terdapat beberapa saran yang dapat dijadikan acuan untuk pengembangan penelitian maupun implementasi lanjutan di lingkungan PT ABC.

Pertama, penelitian ini belum mencakup evaluasi optimasi berbasis konfigurasi parameter database seperti *shared buffers*, work_mem, dan *effective cache size*, karena fokus penelitian diarahkan pada optimasi struktur data melalui *indexing* dan *partitioning*. Oleh karena itu, penelitian selanjutnya disarankan untuk mengombinasikan optimasi struktur dengan *configuration* tuning agar dapat diperoleh gambaran performa yang lebih komprehensif.

Kedua, pengujian pada penelitian ini dilakukan menggunakan data *dummy* yang disesuaikan dengan skema dan distribusi data PT ABC. Untuk pengembangan selanjutnya, disarankan dilakukan pengujian pada data historis yang lebih mendekati beban kerja aktual atau melalui analisis *query* log secara langsung, sehingga hasil evaluasi performa dapat merepresentasikan kondisi operasional secara lebih akurat.

Ketiga, penelitian lanjutan dapat mengeksplorasi teknik optimasi tambahan seperti *query rewriting*, penggunaan *materialized views* untuk laporan periodik, serta pemanfaatan parallel *query execution* secara lebih mendalam. Selain itu, perbandingan performa antar versi PostgreSQL atau pengujian pada *hardware configuration* yang berbeda juga dapat memberikan wawasan tambahan terkait skalabilitas sistem.

Terakhir, dari sisi implementasi, penerapan *indexing* dan *partitioning* pada lingkungan produksi perlu disertai dengan perencanaan yang matang, termasuk pengujian bertahap, pemantauan dampak terhadap operasi *INSERT* dan *UPDATE*, serta mitigasi risiko selama proses *deployment*. Dengan pendekatan yang terkontrol, optimasi database dapat diterapkan secara aman dan berkelanjutan untuk mendukung pertumbuhan data dan kebutuhan sistem PT ABC di masa mendatang.