**BAB II**

**TINJAUAN PUSTAKA DAN DASAR TEORI**

* 1. **Tinjauan Pustaka**

Berikut ini adalah tinjauan pustaka yang digunakan dalam dokumen ini yang mengacu kepada beberapa peneliti yang membuat aplikasi basis data terdistribusi, diantaranya :

1. Karima Al yang membuat penelitian tentang Analisis dan Perancangan Basis Data Terdistribusi Pengolahan Data Polis Asuransi Menggunakan Metode Replikasi Asyncronous pada PT Videi kantor Cabang Palembang pada tahun 2013.
   1. **Dasar Teori**
      1. **Sistem Informasi**

Sistem Informasi adalah suatu kumpulan dari komponen-komponen dalam organisasi yang berhubungan dengan proses penciptaan dan aliran informasi. Pada lingkungan berbasis komputer, sistem informasi menggunakan perangkat keras dan lunak komputer, jaringan telekomunikasi, manajemen basis data, dan berbagai bentuk teknologi informasi yang lain dengan tujuan untuk mengubah sumber data menjadi berbagai macam informasi yang dibutuhkan oleh pemakai (Yakub, 2012).

* + 1. **Basis Data (*Database*)**

Basis data adalah kumpulan data (elementer) yang secara logik berkaitan dalam mempresentasikan fenomena / fakta secara terstruktur dalam domain tertentu untuk mendukung aplikasi dalam sistem tertentu (Fathansyah,2012:2). Sedangkan menurut kadir (2009:41) basis data adalah koleksi dari data-data yang terorganisasi dengan cara sedemikian rupa sehingga mudah dalam disimpan dan dimanipulasi (diperbaharui, dicari, diolah dengan perhitungan-perhitungan tertentu, serta dihapus).

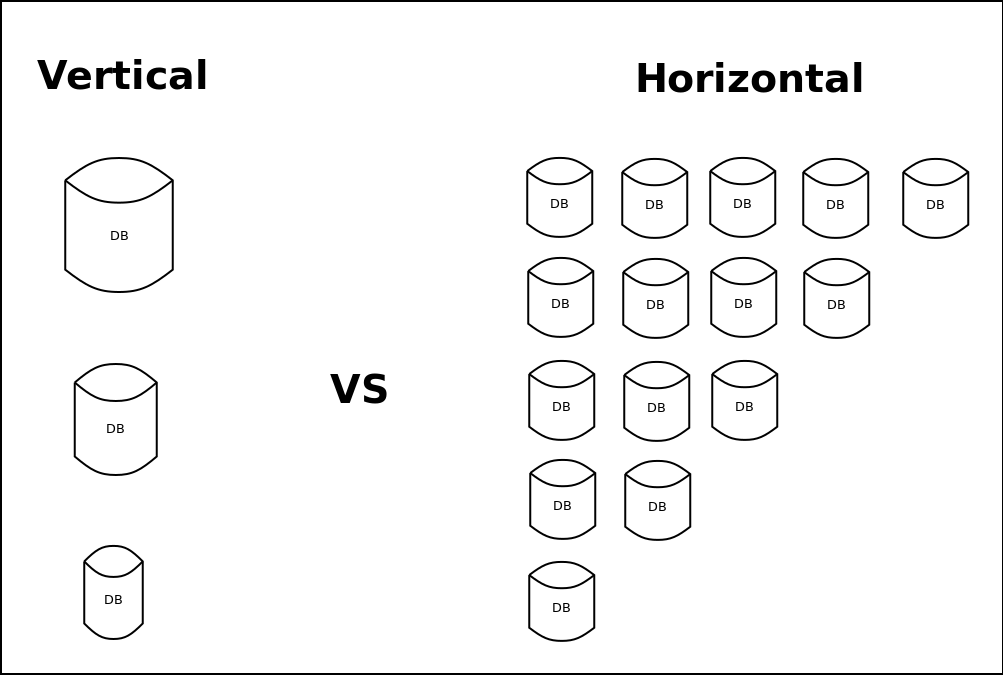
* + 1. **Basis Data Terdistribusi (*Distributed Database*)**

Basis data terdistribusi (*distributed database*) adalah kumpulan data yang digunakan bersama yang saling terhubung secara logic tetapi tersebar secara fisik pada suatu jaringan komputer. (Fathansyah, 2012:335). Karakteristik basis data terdistribusi yaitu :

1. Kumpulan data yang digunakan bersama secara logic tersebar pada sejumlah komputer yang berbeda.
2. Komputer yang dihubungkan menggunakan jaringan komunikasi.
3. Data pada masing-masing situs dapat menangani aplikasi-aplikasi lokal secara otonom.
4. Data pada masing situs di bawah kendali satu DBMS.
   * 1. **CockcroachDB**

*CockroachDB* adalah *key-value datastore* yang mendukung semantik transaksional ACID dan nilai berversi sebagai fitur *first-class*. Tujuan desain utamanya adalah global *consistency* dan *survivability*. *CockroachDB* bertujuan untuk menoleransi *disk*, *machine*, *rack*, dan bahkan kegagalan *datacenter* dengan gangguan *latency* minimal dan tidak ada intervensi manual. *Node CockroachDB* bersifat *symmetric,* Tujuan desain ini adalah penyebaran homogen (satu biner) dengan konfigurasi minimal (Mazumder Suvankar 2015).

Sedangkan menurut Max Wouter Grim (2016) Database secara kasar dapat dibagi menjadi dua jenis: database relasional dan database NoSQL. Konsep database relasional dikembangkan sekitar tahun 1970, sedangkan database NoSQL mulai mendapatkan popularitas sekitar 2009. Saat ini lebih dari 255 database NoSQL berbeda dari jenis yang berbeda ada, termasuk database kunci / nilai, dokumen, grafik dan kolom. Karena jumlah data tumbuh, sistem database harus berskala. CockroachDB dikonfigurasi secara default untuk menyimpan 3 replika datanya. Dalam kasus node crash, data replikasi secara otomatis menyeimbangkan kembali antara node lainnya. Ini memastikan database sangat tersedia. Lokasi baru di cluster diidentifikasi dan replika yang hilang direplikasi ulang secara terdistribusi.



**Gambar 2.1** *Perbandingan skalabilitas basis data.*

* + 1. **NewSQL**

*NewSQL* adalah kelas lain dari sistem manajemen basis data modern. Sampai saat ini implementasi arsitektur *scale-out* membutuhkan beberapa solusi *NoSQL* karena *old-fashioned* basis data relasional tidak memberikan dukungan yang baik untuk skala "horizontal". Seperti yang disebutkan di atas, solusi *NoSQL* semacam itu biasanya tidak memberi ACID dan lebih tepatnya menyediakan beberapa jenis konsistensi akhirnya. Ketegangan inilah yang mengilhami gerakan *NewSQL*.

Solusi *NewSQL* berdasarkan definisi didasarkan pada model relasional. Selain itu, basis data ini berusaha memberikan kinerja sistem *NoSQL* yang sama untuk pemrosesan muatan *online transaction processing* (OLTP), sambil tetap mempertahankan jaminan ACID dari sistem basis data tradisional.

Namun meskipun mereka menggunakan *SQL* sebagai bahasa antarmuka utama dan klien berinteraksi dengan *NewSQL* dalam istilah relasional DB tradisional seperti "tabel" dan "hubungan", representasi internal sebenarnya mungkin sangat berbeda dari DB tradisional. Misalnya, *NuoDB* dapat menyimpan datanya ke dalam *key-value store* (Yuri Gurevich 2015:30).

|  |  |  |  |
| --- | --- | --- | --- |
|  | Old SQL | NoSQL | NewSQL |
| Relational | Ya | Tidak | Ya |
| SQL | Ya | Tidak | Ya |
| ACID Transaction | Ya | Tidak | Ya |
| Hizontal Scalability | Tidak | Ya | Ya |
| Performance / Big Volume | Tidak | Ya | Ya |
| Scheme-less | Tidak | Ya | Tidak |

**Tabel 2.1** *Perbandingan karakteristik basis data.*

* + 1. **Rust**

Rust adalah *systems programming language* bahasa dengan safety code dimana object diatur oleh bahasa pemograman tersebut dari awal hingga akhir. Developer tidak perlu lagi melakukan pointer arithmatic dan manajemen memory seperti yang kita lakukan dalam bahasa C dan C++. Rust juga bahasa dengan general purpose language, bahasa low level yang bisa membantu kita untuk mengeksplorasi potensi sisi system, embedded system, dan hal hal kritis terkait performance. Desain utama dari Rust Programming language adalah fast, efficient and memory safe system. Ini adalah desain yang sudah dijadikan standard modern programming language untuk kedepanya sama seperti yang diterapkan dalam Go Programming Language atau golang (Gun Gun Febrianza 2016).