**SQL vs No SQL**

Sebelum masuk ke perbedaan dari sql dan no sql, saya akan membahas terlebih dahulu mengenai data. Data adalah sebuah bukti nyata dari sebuah peristiwa atau objek yang sudah dikaji. Secara umum, data adalah sebuah bukti yang biasanya berbentuk tulisan atau bentuk yang lain. Dalam pengembangan aplikasi atau software, data sangat dibutuhkan untuk proses pengembangan yang memungkinkan untuk menampilkan data ke user atau meminta data dari user. Data dalam pengembangan software menjadi informasi antar software dengan user. Data biasanya disimpan ke dalam database. Database adalah sebuah tempat penyimpanan data yang digunakan untuk pengembangan software, di dalam database terdapat beberapa dataset yang dibuat untuk mengelompokkan data berdasarkan ketentuan yang sudah dibuat. Dataset yang ada di dalam database biasanya berbentuk tabel, satu tabel ada baris dan kolom dan satu kolom biasanya berisi data tunggal dan satu baris bisanya berisi data tuple. Data tunggal yaitu data yang bersifat satu maksudnya data tersebut hanya berisi satu informasi, misal satu kolom data tunggal adalah kolom nama, maka data tunggal dari kolom tersebut adalah masing – masing nama yang terdapat pada kolom tersebut. Sedangkan data tuple adalah data yang berisi satu set data, misal dalam satu tabel terdiri dari kolom nama, Alamat, dan nomor hp, maka data tuple dari satu set data adalah satu set informasi dari satu baris, contoh nama = aan, Alamat = undaan, nomor hp = 329.

Pada pengembangan software, terdapat 2 istilah yang sering digunakan untuk menyimpan data, yaitu SQL dan No SQL. Keduanya digunakan dengan studi kasus yang berbeda. No SQL digunakan jika:

1. Basis data relasional tidak restriktif.

Ketika kita ingin mengembangkan software yang mana awalnya memang software tersebut tidak menyimpan data yang terdistribusi atau data yang bervariasi, Ketika software yang kita kembangkan sudah menjadi software yang besar, maka akan banyak variasi data yang harus kita simpan ke dalam system kita, Ketika hal tersebut terjadi maka saatnya kita mengubah system database kita dari SQL menjadi NoSQL. Jadi, **NoSQL digunakan untuk menyimpan data terdistribusi atau data yang mempunyai variasi yang banyak.**

1. Ketika ACID *(Atomicity Consistency Isolation Durability)* sudah tidak dibutuhkan.

Dalam konteks pemrosesan transaksi, terdapat ACID yang mengacu pada empat sifat utama transaksi.

1. **Atomicity**

Semua perubahan data dilakukan seolah-olah itu adalah satu operasi. Misal semua transaksi terjadi atau tidak sama sekali. Jadi istilahnya kalo dijalnkan satu ya jalan semua kalo tidak dijalankan ya tidak dijalankan semua. Jika kita ingin memindahkan uang dari Bank A ke Bank B, lalu terjadi error pada server Bank B padahal kita sudah memencet tombol kirim, apa yang terjadi? Jika system bank A dan Bank B tidak menggunakan Atomicity, maka uang tersebut akan hilang, Atomicity mencegah terjadinya hal tersebut. Maka dari itu istilahnya jika dijalankan ya jalan semua, jika dijalankan ke server Bank B ya dijalankan keduanya Bank A dan Bank B, jika tidak dijalankan ya tidak jalan semua, jika dijalankan ke Bank B ternyata server Bank B mengalami error, maka transaksi ke Bank B tidak dapat dijalankan dan perpindahan uang dari Bank A ke Bank B pun dibatalkan. Jika server error, maka system akan melakukan perintah yang Namanya *“Rollback”*, maka keduanya dibatalkan, jika server atau system tidak mengalami error, maka system akan melakukan perintah yang Namanya *“Commit”*, maka keduanya dapat dijalankan secara lancar. Atomicity juga bersifat keterkaitan.

1. **Consistency**

Consistency digunakan untuk memeriksa apakah setiap operasi transaksi mematuhi Batasan integritas data dan aturan bisnis yang telah ditetapkan dalam transaksi basis data. Misal terdapat nasabah A yang ingin menarik uangnya sebesar 1.000.000, consistency memastikan data berperilaku konsisten, jika consistency diterapkan maka system akan mengecek dari database apakah penarikan 1.000.000 melebihi nominal rekening nasabah A yang terdapat di dalam database. Misal nominal saldo nasabah A yang terdapat di dalam database adalah 500.000, maka prinsip consistency di dalam system program aplikasi maupun di dalam system database akan menolak atau *“Rollback”* atau membatalkan proses yang sedang dilakukan, jika prinsip consistency tidak diterapkan proses akan tetap dilakukan dan nominal saldo nasabah A yang ada di dalam database berubah menjadi ( -500.000 ). Prinsip consistency adalah aturan data yang bersifat relative yang menyesuaikan aturan dari bisnis atau diri sendiri yang telah ditetapkan.

1. **Isolation**

Isolation mengacu pada bagaimana transaksi data dalam database harus diisolasi antara satu sama lain. Singkatnya database harus bisa mengatur atau mengisolasi perintah yang akan dijalankan untuk mengubah, menulis, atau menghapus data pada setiap perintah. Misal suatu website menjual barang dengan stok yang terbatas yaitu 100 stok, jika ada pembeli 1 yang ingin membeli 100 stok dan ada pembeli 2 yang membeli 20 stok dalam waktu yang sama. Jika website tersebut tidak menggunakan prinsip isolation maka kedua transaksi tersebut akan dieksekusi dan membuat nilai stok yang ada di database menjadi minus menjadi (-20). Jika menggunakan prinsip consistency biasanya hanya satu transaksi yang akan dijalankan jika ada transaksi dalam waktu yang sama. Istilah ini dikenal sebagai *“Serial Execution”*. Misal transaksi yang dijalankan dulu adalah transaksi pembeli 1 yang membeli 100 stok, prinsip consistency juga dapat mengecek ketersediaan stok yang ada di dalam database, maka transaksi dari pembeli 1 dapat berjalan dengan lancar, lalu transaksi pembeli 2 yang membeli 20 stok akan dieksekusi, sebelum dieksekusi akan dicek terleih dahulu ketersediaan stok yang ada di dalam database, jika stoknya 0, maka transaksi akan dibatalkan.

1. **Durability**

Durability menjamin bahwa sekali transaksi telah disimpan, itu akan tetap disimpan bahkan dalam kasus kegagalan system (misalnya, pemadaman listrik atau crash). Misal ada pelanggan yang mau checkout pesanannya dalam sebuah website, jika tombol checkout sudah ditekan maka system akan menganggap transaksi data tersebut sudah di commit dan disimpan ke dalam database. Jika terdapat error pada system seperti mati listrik dan kendala yang lain, data tersebut akan tetap ada karena sudah di commit sebelum ada error pada system. Jadi data akan tetap aman jika sudah di commit dan tidak terganggu karena adanya error pada system, apabila error pada system terjadi saat transaksi data belum di commit, maka data tersebut tidak akan ke record ke dalam database.

Ketika ACID sudah tidak dibutuhkan, system yang tidak memenuhi kriteria ACID disebut dengan BASE, yaitu Basically Available, Soft State, dan Eventual Consistent. Sebagian besar NoSQL saat ini adalah BASE.

1. Adanya hambatan antar objek untuk berelasi

Ketika sebuah studi kasus yang menginginkan relasi antar objek dan relasi tersebut terhambat karena adanya banyak factor, maka studi kasus tersebut seharusnya menggunakan NoSQL. Jadi NoSQL tidak menggunakan skema relasional yang ketat dan struktur dari NoSQL lebih fleksibel dibanding dengan strkutur SQL.

1. RDBMS (Relational Database Management System) sudah tidak berdistribusi dan scalable

Secara umum RDBMS tidak dirancang secara alami untu menjadi system yang terdistribusi dan scalable secara horizontal. Karena RDBMS tidak secara otomatis didistribusikan ke beberapa node atau server, hal tersebut bertentangan dengan konsep dari terdistribusi, karena RDBMS hanya mengandalkan satu node atau server saja dan jika satu node atau server tersebut mengalami kegagalan maka aplikasi yang terhubung ke database ini akan mengalami gangguan atau kehilangan akses datanya. RDBMS juga mempunyai keterbatasan dalam menambahkan node atau server ke dalam cluster. Hal tersebut dikarenakan tabel dan relasi data yang rumit. Sebenarnya bisa bisa saja RDBMS menambahkan node atau server ke dalam cluster, namun jka ingin melakukan hal tersebut kita harus mengatur partisi data dan menjaga konsistensi data sebelum di distribusikan ke node atau server yang baru. Ketika ingin melakukan partisi data pada RDBMS, hal yang paling menghambat dalam melakukan partisi adalah kompleksitas relasi tabelnya, sehingga kita harus memikirkan bagaimana caranya agar data yang terdistribusi dari satu node ke node yang lainnya dapat berhubungan dengan baik dan dapat mempertahankan consistency datanya. Dari hal tersebut memang memungkinkan untuk RDBMS di distribusikan tapi cara pendistribusiannya dan menjaga data agar tetap konsisten, bahkan setelah melakukan distribusi, ada Batasan sejauh mana kita dapat meningkatkan kinerja dengan menambahkan node. Hal tersebut menjadi tidak efisien dan disarankan menggunakan NoSQL saja.

1. Dipakai Ketika kita menggunakan data yang bersifat temporal (sementara)

Contoh dari data temporal adalah Shopping Cart, Whistlist dan lain lain. Shopping cart dan Whistlist akan hilang secara otomatis Ketika stok di dalam database sudah habis.