Optimasi Schema dan Tipe Data

Herdi Ashaury

Topik

Pemilihan Tipe Data

Design Schema dalam MySQL

Capaian Pembelajaran

Mahasiswa mampu merancang optimalisasi schema dan tipe data dari database

Tipe Data dan Database

MySQL mempunyai banyak tipe data.

Memilih tipe data yang tepat untuk menyimpan data sangat penting untuk mendapatkan performansi yang bagus.

Pemilihan Tipe Data

Smaller is usually better, semakin tipe data yang digunakan kecil maka akan semakin cepat, karena penggunaan dihardisk, memory dan cache juga sedikit. Tentukan Range Value yang bisa disimpan.

Simple is good, sebagai contoh, tipe data integer lebih sederhana dari char (varchar), karena char mempunyai char sets dan collations.

Avoid NULL if possible, dbms kesulitan dalam mengoptimasi query yang bernilai NULL.

Numbers (Integer)

Jenis integer, TINYINT, SMALLINY, MEDIUMINT, INT atau BIGINT

Banyak bit, 8, 16, 24, 32 dan 64

Value $-2^{(N-1)}$ sampai $2^{(N-1)}-1$ dimana N adalah jumlah bit

Gunakan UNSIGNED untuk menambah range jika memang tidak ada nilai negative sehingga dari pada -128 sampai 127, menjadi 0 sampai 255 (untuk kasus 8 bit)

Beberapa aplikasi tidak memeperhatikan range tipe integer, missal INT(1) sama dengan INT(20)

Numbers (Real)

Bisa digunakan untuk Integer yang terlalu besar melebihi kapasitas BIG INT.

Untuk Floating point, batasi jumlah digitnya, contoh DECIMAL (5,2) akan menyimpan dua digit untuk setiap sisi dibelakang koma.

Ukuran Real Number : DECIMAL > DOUBLE > FLOAT

DECIMAL gunakan jika hanya membutuhkan hasil yang presisi. (ex: data keuangan).

String (Varchar dan Char)

VARCHAR

- Variabel-length (less space)
- disimpan sebagai VARBINARY (character)
- Cocok untuk tipe string yang panjang nilainya berbeda-beda.
- Performansi kurang baik untuk proses update

CHAR

- Fixed-length (space memory tetap diallocate meskipun tidak terpakai semua)
- disimpan sebagai BINARY (bytes)
- Cocok untuk tipe string yang panjang nilainya tetap (ex. MD5)
- Performansi baik untuk proses update

BLOB dan Text

BLOB dan TEXT merupakan string yang didesain untuk menyimpan data besar berupa binary atau character.

MySQL handle BLOB dan TEXT sebagai object.

BLOB menyimpan tanpa collation dan character set

TEXT menyimpan dengan collation dan character set

MySQL Sort berdasarkan panjang nilai

Tidak bisa digunakan sebagai Index

ENUM

Besar bytes yang disimpan tergantung jumlah pilihan atau list dalam ENUM

ENUM disimpan seperti suatu table look up dan satu integer sebagai posisi dari list

Jangan gunakan ENUM untuk bilangan konstan (ex. 1,2,3,…)

Lebih cepat dalam proses join dibandingkan VARCHAR

Datetime dan Timestamp

DATETIME

- Dapat menyimpan range yang besar (Tahun 1001 – 9999)
- Tidak bergantung pada timezone
- Disimpan sebagai integer dengan format YYYMMDDHHMMSS
- 8 bytes

TIMESTAMP

- Menyimpan nilai dari tahun 1970 - 2038
- Bergantung pada timezone
- Nilai 0 sama dengan 1969-12-31 19:00:00 (EST)
- 4 bytes
- Default set dengan nilai current time jika di insert atau update tanpa nilai.

Kesalahan Umum Pada Suatu Schema

Terlalu Banyak Kolom,

MySQL bekerja dengan mengcopy tiap baris lalu server mendecode buffer tersebut menjadi kolom-kolom. Sehingga dapat memberatkan server jika ada banyak kolom.

Too many joins, limit MySQL adalah 61 tables per join.

Overusing ENUM, contoh salah enum(0,1,2,3,….,10).

The ENUM in disguise, penggunaan SET dapat digunakan seperti ENUM

Null not invented here, penggunaan NULL memang tidak dianjurkan, tetapi jika memang diperlukan tidak perlu menggunakan suatu nilai konstan. Contoh nilai 0000-00-00 untuk date.

Normalisasi

Proses Update lebih cepat

Tidak ada / minim duplikasi data

Ukuran Tabel kecil

Mengurangi penggunaan DISTINCT ata GROUP BY

Banyak membutuhkan query join (Sangat jelek untuk performansi)

Strategi indexing tidak bisa diterapkan

Denormalisasi

Tidak memerlukan join

Harus Full Table Scan

Strategi Indexing lebih mudah dan efisien

Normalized or Denormalized?

In Real world, perlu mencampurkan keduanya

Menambahkan trigger untuk melengkapi proses update

Menambahkan field tambahan selain foreign key pada tabel yang sering dilakukan select untuk mengurangi join

Buat “Cache Table” dan “Summary Table”

Materialized Views

View yang sudah diproses sebelumnya dan tersimpan dalam disk, dan direfresh secara berkala sesuai dengan strategi.

MySQL tidak memiliki materialized views, tapi bisa dibuat dengan cara yang lain.

Salah satu tools menggunakan Flexviews.

Kesimpulan

Desain Schema yang baik sangat relatif.

Lebih baik menjaga segala sesuatu sekecil dan sederhana mungkin

Hindari schema yang dapat mengakibatkan query yang kompleks, atau tabel yang terlalu besar dan banyak kolom.

Hindari nilai NULL kecuali memang dibutuhkan

Gunakan tipe data yang sama untuk nilai yang sejenis atau terkait.

Perhatikan panjang variabel pada penggunaan string, dapat mengakibatkan alokasi memori yang besar dan penyortiran yang sulit.

Gunakan bilangan bulat untuk identifier (PRIMARY KEY)

Gunakan ENUM dan SET dengan tepat

Normalisasi dan denormalisasi bermanfaat dan diperlukan.

Referensi

Baron Schwartz, Peter Zaitsev, Vadim Tkachenko - High performance MySQL_ optimization, backups, and replication-O'Reilly Media (2012)

Terima Kasih