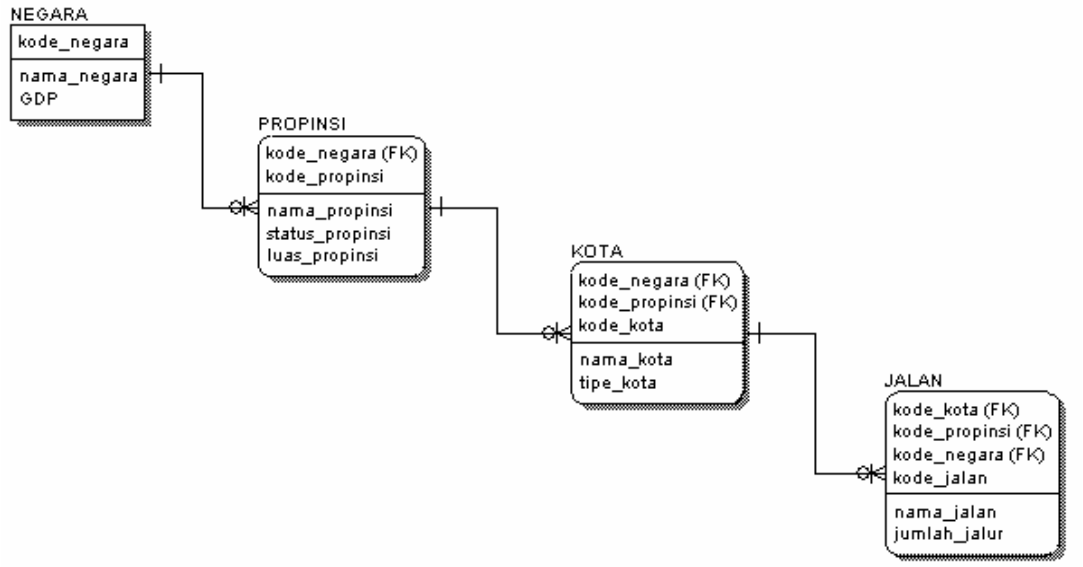
Surrogate Key

Djoni Darmawikarta
djoni_darmawikarta@yahoo.ca

Misalkan kita memiliki database *alamat* dengan diagram ER (Entity Relationship Diagram) data model-nya sebagai berikut. Perhatikan bahwa attribute yang didalam bagian atas dari kotak entity adalah primary key. Misalnya untuk entity NEGARA, primary key adalah kode_negara. Sedang PROPINSI memiliki composite primary key yaitu kode_negara dan kode_propinsi.



Dalam model ini primary key tabel NEGARA, kode_negara, menjadi bagian primary key tabel PROPINSI; lalu menjadi bagian dari primary key tabel KOTA, dan juga bagian primary key tabel JALAN. Demikian juga dengan primary key tabel ditingkat lebih atas yang lain, menjadi bagian dari primary dari primary key tabel yang dibawahnya, dan dibawa terus ke yang lebih bawah. Dengan demikian dapat dipastikan

jalan yang dinegara, dikota, dan dipropinsi mana, karena mungkin ada jalan yang sama dinegera, kota dan propinsi berbeda.

Dalam contoh model ini, kode_negara, propinsi, kota, dan jalan dibentuk berdasarkan namanya, seperti dapat dilihat sample data JALAN dibawah ini.

Kode_negara	Kode_propinsi	Kode_kota	Kode_jalan	Nama_jalan	Jumlah_ jalur
IND	JABAR	JAK	PDHJ	Pondok Hijau	2
IND	JABAR	JAK	PNID	Pinang Indah	2

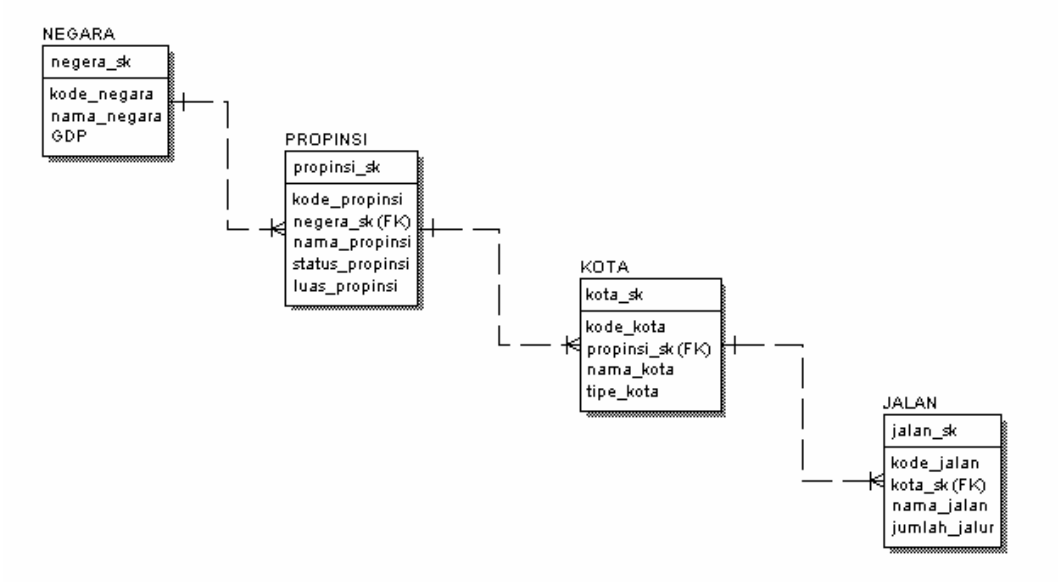
Apa yang terjadi bila terjadi penggantian nama, misalnya Jawa Barat, dengan kode_propinsi JABAR, menjadi Daerah Istimewa Jakarta Raya, yang menurut aturan pemberian kode, kode_propinsi-nya juga harus diganti, misalnya menjadi DIJR.

Semua data isi tabel JALAN, yang berkaitan dengan Jawa Barat, harus diubah! Demikian juga dengan isi tabel KOTA. Rangkaian perubahan ini makin memburuk bila PROPINSI, KOTA dan/atau JALAN digunakan (dihubungkan) dengan tabel-tabel data yang lain.

Maslah ini terjadi karena rancangan diatas menggunakan intelligent key (bermakna); kode_negara, propinsi, kode dan jalan, yang menjadi primary key, dibentuk (dirumuskan) berdasarkan namanya. Intelligent key kadang disebut juga “natural” key.

Solusi Surrogate Key

Surrogate key adalah solusi yang efektif. Surrogate key, sebagai primary key, yang biasanya merupakan angka bulat urut yang tidak bermakna, tidak ada hubungan dengan data didalam tabel. Dengan surrogate key, diagram ER kita menjadi sebagai berikut.



Kini, bila nama propinsi dan kodenya berubah, atau attribute yang manapun, data ditabel KOTA dan JALAN, tidak kena dampaknya.

Bila kita ingin menyimpan nama dan kode sebelumnya (history), kita menambahkan record baru yang memiliki surrogate key-nya sendiri, tidak menimpanya dengan yang baru. Teknik surrogate key untuk menyimpan data historis ini, didunia data warehousing dikenal dengan istilah "slowly changing dimension type two".

Fasilitas Surrogate Key

Database populer seperti Oracle, DB2 dan SQL Server memiliki fasilitas implementasi surrogate key. Berikut contoh penggunaannya di Microsoft SQL Server.

Mendefinisikan surrogate key pada waktu membuat tabel. Misalnya untuk tabel NEGARA bisa menggunakan SQL statement sebagai berikut.

```
CREATE TABLE negara
        ( negara_sk int IDENTITY(1,2) PRIMARY KEY,
        kode_negara varchar(10) NOT NULL,
        nama_negara varchar(50) NOT NULL,
        gdp int)
```

Perhatikan tambahan IDENTITY(1,2) pada surrogate key negara_sk. Angka pertama (disebut *seed*) adalah nilai negara_sk yang akan diberikan oleh SQL Server untuk data (row) pertama yang diisikan kedalam tabel ini, dalam contoh kita adalah 1. Angka kedua (disebut *increment*), yaitu 2 dalam contoh kita ini, adalah pertambahan pada nilai surrogate key dari nilai terakhir yang sudah dipakai untuk mendapatkan nilai yang akan diberikan kepada data yang sedang dimasukkan.

Bila kita jalankan tiga buah SQL statement berikut, setelah tabel dibuat (masih kosong datanya)

```
INSERT negara (kode_negara, nama_negara, gdp)
        VALUES ('IND', 'Indonesia', 100)
INSERT negara (kode_negara, nama_negara, gdp)
        VALUES ('SGP', 'Singapura', 1000)
INSERT negara (kode_negara, nama_negara, gdp)
        VALUES ('KAN', 'Kanada', 1100)
```

Maka isi tabel negara adalah:

negara_sk	kode_negara	nama_negara	gdp
1	IND	Indonesia	100
3	SGP	Singapura	1000
5	KAN	Kanada	1100

Membaca data, misalnya dengan SQL sebagai berikut:

```
SELECT nama_negara, gdp FROM negara
        WHERE gdp > 100
```

tidak perlu ‘menyentuh’ surrogate-key, karena memang dia, dilihat dari sudut user (bukan technical/system), tidak ada hubungan dengan data.

Untuk query yang melibatkan dua atau lebih tabel (join query), misalnya SQL berikut:

```
SELECT a.nama_negera, b.nama_propinsi FROM negara a, propinsi b
        WHERE a.kode_negara = ‘IND’
                AND a.negara_sk = b.negara_sk
```

kita tidak perlu menyentuh isi surrogate key, cukup ditambahkan bagian terakhir yang menyebutkan surrogate key dari kedua tabel yang dibutuhkan untuk menghubungkan mereka.

Dapat disimpulkan, bahwa penggunaan surrogate key adalah ‘transparan’ terhadap user.

Kesimpulan Manfaat Surrogate Key

Selain menjamin kestabilan (tabel terkait tidak kena dampak perubahan), surrogate key juga menghemat tempat penyimpan (disk space), karena hanya satu attribute numerik sederhana, tidak composite seperti bila digunakan intelligent key (makin panjang rangkaian entity-nya makin besar composite primary key-nya entity ujung terbawahnya, dan dia makin tidak stabil)

Surrogate key yang sering disebut dummy key sebenarnya smart! Maksudnya, dia tidak bermakna dan tidak perlu dipusingkan – dia otomatis berfungsi memecahkan masalah intelligent key.