



User Registration and Management

Erick Kurniawan



Intro

- Pada bab sebelumnya “Software Structure” dikatakan:
- Semakin banyak user yang diidentifikasi dan diautentikasi, untuk membangun OLC yang sukses
- Karena itu user database seharusnya menyimpan informasi sebanyak mungkin yang dapat membantu user A mengenal user B

Ide : Fat vs Skinny

- Misalnya untuk spesifikasi sistem awal anda membuat desain table user yang simpel:

```
create table users (  
  user_id integer primary key,  
  first_names varchar(50),  
  last_name varchar(50) not null,  
  email varchar(100) not null unique,  
  --mengkrip passwor dengan fungsi  
  crypt function password varchar(30)  
  not null, registration_date  
  timestamp(0) );
```

- Penting untuk melakukan enkripsi pada password dengan fungsi yang disediakan oleh DBMS

Ide : Fat vs Skinny

- Tips. bagus untuk membiasakan diri mendesain tabel pada text editor dan menambahkan komentar, daripada langsung menggunakan GDI
- Setelah beberapa minggu online mungkin ada yang menyarankan untuk menambah fasilitas foto dan link ke url lain

```
create table users ( user_id integer
primary key, first_names varchar(50),
last_name varchar(50) not null, email
varchar(100) not null unique, password
varchar(30) not null, url varchar(200),
registration_date timestamp(0),
-- menggunakan tipe data BLOB
portrait blob );
```

Ide : Fat vs Skinny

- Setelah beberapa bulan....

```
create table users ( user_id integer primary key,  
first_names varchar(50), last_name varchar(50) not  
null, email varchar(100) not null unique, password  
varchar(30) not null, url varchar(200),  
registration_date timestamp(0) -- use the image  
datatype instead of BLOB portrait blob,  
-- dengan maksimum 4 GB  
biography clob, birthdate date,  
sex char(1) check (sex in ('m','f')), country_code  
char(2) references country codes(iso), postal code  
varchar(80), home_phone varchar(100), work_phone  
varchar(100), mobile_phone varchar(100), pager  
varchar(100), fax varchar(100), aim_screen_name  
varchar(50), icq_number varchar(50) );
```



Ide : Fat vs Skinny

- Tablanya semakin bertambah kompleks (“fatter”)
- Banyak column yang akan bernilai null untuk setiap record user
- Oracle 9i juga mempunyai batasan limit 1000 column per-table (mengatasi storage efficiency problem)
- Untuk setiap nilai null DBMS hanya memerlukan 1 bit saja
- Pada Medical Information System sudah lama mengalami masalah ini “fat data model” (misal: untuk tes lab, ada ribuan kemungkinan jenis tes)

Desain Table

- Ide desain tablenya sebagai berikut:
- Dipecah menjadi dua table: skinny table dan table kedua

```
create table users
(
user_id integer primary key,
first_names varchar(50),
last_name varchar(50) not null,
email varchar(100) not null unique,
password varchar(30) not null,
registration_date timestamp(0)
);
```



Desain Table

```
create table users_extra_info (  
user_info_id integer primary key,  
user_id not null references users,  
field_name varchar(100) not null,  
field_type varchar(100) not null,  
-- one of the three columns below will be non-NULL  
varchar_value varchar(4000),  
blob_value blob,  
date_value timestamp(0),  
check ( not (varchar_value is null and blob_value is  
null and date_value is null))  
-- in a real system, you'd probably have additional  
columns to store when each row was inserted and by whom  
);  
-- mencari informasi untuk user tertentu secara cepat  
create index users_extra_info_by_user on  
users_extra_info(user_id);
```

Desain Table

user_id	first_names	last_name	email	password
1	Wile E.	Coyote	supergenius@yahoo.com	IFUx42bQzgMjE

user_info_id	user_id	field_name	field_type	varchar_value	blob_value	date_value
1	1	birthdate	date	--	--	1949-09-17
2	1	biography	blob_text	--	Created by Chuck Jones...	--
3	1	aim_screen_name	string	iq207	--	--
4	1	annual_income	number	35000	--	--

Hal yg perlu diperhatikan

- Misal: ada permintaan untuk menampilkan rata-rata annual_income semua user yang terdaftar, oracle dapat secara otomatis membedakan number dan string

```
select average(varchar_value) from users_extra_info
where field_name = 'annual_income'
```

- Contoh: tidak mungkin user yang sama memiliki user_id dan field_name yang sama

```
create unique index
users_extra_info_user_id_field_idx on
users_extra_info (user_id, field_name);
```

- Membuat pencarian field_name untuk user_id menjadi lebih cepat



Hal yg perlu diperhatikan

- Bagaimana dengan home phone? Mis: satu orang mempunyai beberapa nomor telepon
- Memasukan home phone dengan user yang sama akan menyebabkan violation terhadap unique constraint
- Caranya dengan menambahkan logic pada program (misal: cek adanya duplicated value, dua home phone atau birthday untuk user yang sama, jika ditemukan munculkan pesan)
- Atau menggunakan trigger yang dapat menjalankan procedural program PL/SQL secara otomatis



Fat vs Skinny: Decision

- Disesuaikan dengan permasalahan yang ada (tergantung programmer) kita bisa men-develop aplikasi dengan kedua cara tersebut
- Fat-style lebih mudah dimaintain dan self-documentation
- Fat-style umum digunakan pada database, biasanya menggunakan data dictionary
- Programmer yang mengambil alih pekerjaan anda akan lebih senang dengan fat-style karena lebih mudah dimengerti
- Untuk table yang kolomnya bisa bertambah terus fat-style tidak cocok, terlalu banyak field yang bernilai null, ada batasan kolom per-table



Fat vs Skinny: Decision

- Skinny bagus digunakan jika anda menyimpan data yang berlainan setiap user (banyak column tambahan)
- Jika desain menggunakan fat-style ternyata banyak nilai null pada data (kira-kira 75%) maka sebaiknya menggunakan skinny-style
- Skinny mempunyai desain yang terlihat aneh / tidak biasa
- Data kurang jelas dimengerti (hanya programmer yang mendesain database yang paham)



User Groups

- Salah satu fasilitas yang penting dalam OLC adalah user group
- Group dari user mungkin menginginkan diskusi special topik yang mereka senangi, atau ingin membuat private discussion forum
- Desain-nya terdiri dari 2 table yaitu table USER dan table USER_GROUP yang masing-masing memiliki fields user_id dan user_group_id yang nilainya tidak null

User Groups: (desain table 1NF)

- Misalkan anda punya table USER dan USER_GROUPS bagaimana menyimpan data “user 234 adalah anggota dari group 17 dan 18”
- Cara yang paling simpel mungkin anda dapat menyimpannya di table USER

```
create table users
(
  user_id integer primary key, ...
  -- a space-separated list of group IDs
  group_memberships varchar(4000), ...
);
```



User Groups: (desain table 1NF)

- Anda dapat menyimpan string “17 18” pada kolom group_membership
- Hal ini tidak boleh dilakukan (repeating group atau multivalued column), masalahnya:
- Anda mungkin tidak punya banyak space jika kolom bertambah besar dari yang diperkirakan
- Perintah insert, update, delete tidak mampu untuk memanipulasi multivalued column
- Harus dalam bentuk 1NF (First Normal Form)

User Groups: (desain table 1NF)

- Solusinya adalah membuat table menjadi 1NF yang tidak mengandung multivalued column

```
create table user_group_map
(
user_id not null references users;
user_group_id not null references
user_groups;
  unique(user_id, user_group_id)
);
```

- Many-to-One relationship



Derivable Data

- Memisahkan user dan group dalam 3 tabel, bagaimana cara mengambil informasinya? Misal: “berada di group manakah user erick kurniawan”
- Kita harus melakukan JOIN untuk ketiga table tersebut

```
select user_groups.group_name from users,  
       user_groups, user_group_map where  
       users.first_names = 'erick' and users.last_name  
       = 'kurniawan' and users.user_id =  
       user_group_map.user_id and  
       user_groups.user_group_id =  
       user_group_map.user_group_id;
```

Derivable Data

- Mis: Apakah erick kurniawan adalah anggota dari group csharp

```
select count(*) from user_group_map where user_id
= (select user_id from users where first_names
= 'erick' and last_name = 'kurniawan') and
user_group_id = (select user_group_id from
user_groups where group_name = 'csharp')
```

- Jika groupnya sangat populer kita bisamelakukan denormalisasi data model dengan menambahkan pada user table kolom csharp_group_member (nilainya hanya t & f)
- Jika ingin mempermudah penggunaan query, gunakan VIEW



Derivable Data

```
create view csharp_group_members as select * from
  users where exists (select 1 from
  user_group_map, user_groups where
  user_group_map.user_id = users.user_id and
  user_group_map.user_group_id =
  user_groups.user_group_id and group_name =
  'csharp' );
```

- Clean up ugly query dengan view
- Clean up ugly performance problem dengan index

Access Control & Approval

