

Database Embedded dengan SQLite

Untuk kebutuhan database, tidak semua pengguna harus menggunakan MySQL ataupun PostgreSQL. Keduanya membutuhkan *resource* sistem yang cukup besar dan sistem databasenya pun cukup rumit. Apabila Anda membutuhkan database “satu file”, yang kecil, sederhana, bisa dipindahkan dengan sangat mudah, maka Anda selalu bisa mencoba SQLite.

Database *embedded* selalu menjadi cerita yang seru bagi pengguna yang ingin segalanya serba praktis. Dan, jumlah pengguna tersebut sangatlah banyak di seluruh dunia. Penulis juga merupakan salah satunya. Tak heran apabila database semacam DB(F) dan MDB sangat populer sampai hari ini. Berjuta-juta pengguna komputer di dunia ini masih menggunakan database jenis tersebut. Dengan database *embedded*, semua data disimpan di dalam—umumnya—satu file tunggal, yang menjadikannya sangat mudah untuk dibagi, sangat mudah untuk dipindahkan dan secara umum, sangat mudah untuk di-*manage*.

Database system semacam ini jelas bertentangan dengan database yang bekerja secara client/server seperti PostgreSQL ataupun MySQL. Arsitektur database system keduanya memungkinkan adanya bagian yang jelas antara server database dan client database. Dengan demikian, untuk kebutuhan data diakses ramai-ramai, pengguna tidak perlu mengatur file sharing secara manual, karena database system-nya sendiri telah mengizinkan. Dibandingkan dengan database *embedded*, database semacam ini jelas memiliki keunggulan seperti multi user native seperti disebutkan sebelumnya, skema keamanan yang lebih fleksibel, dan umumnya, lebih mendukung sistem *enterprise*. Namun, jangan lupa bahwa proses replikasi, *back-up*, dan *restore* juga tidak mudah.

Sebagai pengguna database, pemilihan suatu database system merupakan hal

yang sangat penting. Benar bahwa hampir semuanya menggunakan bahasa SQL standar, namun, tidak semua menuruti standar benar-benar. Boleh dikatakan, masing-masing database system yang menerapkan SQL pun, umumnya menambahkan fitur sendiri yang oleh karenanya tidak dimiliki oleh database system lain. Lebih detail, ada pula masalah tipe data yang disebabkan tidak menu-ruti standar SQL.

Untuk memilih database system itu sendiri, Anda mungkin akan memilih dulu apakah database system yang akan Anda gunakan tersebut termasuk kategori yang *embedded* atau yang bekerja secara client/server. Berikut ini adalah beberapa alasan untuk menggunakan database *embedded*:

- Anda akan sering memindah-mindahkan database dari satu tempat ke tempat lainnya.
- Database tidak digunakan untuk melayani kebutuhan *enterprise* (contoh: tidak perlu database clustering).
- Anda tidak membutuhkan banyak fitur *advanced* (contoh: tidak membutuhkan trigger, stored procedure, dan lain-lain).

Apabila Anda memiliki alasan-alasan tersebut, maka tidak ada salahnya apabila Anda mencoba menggunakan *embedded* database. Bicara soal ukuran data yang mampu ditampung oleh database *embedded*, umumnya hal tersebut bukan merupakan masalah lagi. Ukuran yang mampu ditampung oleh suatu *embedded* database juga tergantung pada sistem operasi yang digunakan. Apabila diperkirakan data-

base Anda perlu menampung database berukuran sekitar 2 GB misalnya, maka banyak sekali database *embedded* yang sudah mendukung. Ukuran file 2 GB juga bukan masalah besar untuk sistem operasi modern. Beberapa database *embedded* secara teori mampu menampung data sampai satuan Tera, namun, apabila data Anda sudah berkisar pada ukuran tersebut, akan sangat disarankan apabila Anda menggunakan database system yang memiliki dukungan untuk dunia *enterprise*.

Mari kita batasi bahwa kita akan menggunakan database *embedded*. Solusi tidak serta merta muncul begitu saja karena terdapat begitu banyak database *embedded* yang ada di pasaran. Untuk di dunia *open source* saja, diperkirakan ada belasan database *embedded* yang tersedia. Walau, yang benar-benar bisa diandalkan barangkali hanya di bawah sepuluh. Beberapa yang sangat populer diantaranya DB dari Sleepycat dan SQLite.

Bicara soal database, umumnya yang selalu terbayang di dalam benak kita adalah sintaks SQL. Sayangnya, tidak semua database *embedded* mendukung fitur yang satu ini. Banyak sekali *embedded* database yang hanya mendukung operasi pencarian dengan *lookup* pada key, sementara operasi penambahan data dan modifikasi atau penghapusan data menggunakan API yang telah disediakan. DB (BerkeleyDB) misalnya. Walaupun merupakan salah satu database *embedded* terpopuler yang ada, tidaklah mendukung sintaks SQL. Keputusan untuk tidak datang bersama fitur ini umumnya macam-macam. Ada yang memang karena

keterbatasan teknologi, ada yang dikarenakan tujuan optimasi dan compactness. Dengan tidak datang bersama fitur dukungan SQL, maka setidaknya tidak akan ada kegiatan parsing dan mengartikan sintaks SQL yang diberikan. Kecepatan proses pun bisa ditingkatkan.

Mengenal SQLite

Dari beberapa embedded database yang memiliki cukup banyak fitur, sekaligus datang dengan kemampuan SQL, penulis cukup senang dengan kehadiran SQLite. Ada beberapa alasan mengapa SQLite menyenangkan:

- Secara umum sangat stabil. Penulis cukup lama menggunakan SQLite dan tidak ada masalah yang berarti.
- Lisensi yang sangat menyenangkan. SQLite merupakan software yang dilisensikan *public domain*. Ini jauh lebih menyenangkan daripada GPL.
- Sangat cepat.
- Dari sisi dukungan SQL, mendukung banyak sekali standar SQL92.
- Mampu menampung data sampai 2 Tera. Walaupun yang satu ini umumnya tidak terlalu menjadi fitur yang paling menarik, setidaknya, SQLite mampu menampung data sangat besar.
- Tidak memerlukan banyak memory. Ini sangat berguna untuk sistem yang juga embedded.
- Banyak bahasa program telah mendukung database yang satu ini.
- Mendukung ACID, bahkan pada kegagalan sistem atau power.
- Perkembangannya juga cukup stabil.
- Dan, tentunya, sebagai database embedded, tidak ada usaha tambahan untuk pengaturan data, selain itu, database pun disimpan dalam satu file tunggal. File tersebut dapat dipindahkan bahkan dari mesin LE (Little Endian) ke BE (Big Endian) dan lain sebagainya.
- Tabel dapat disimpan pada file terpisah, dan dapat di-attach ke database utama.

SQLite adalah database system yang dikembangkan oleh D. Richard Hipp dari HWACI Applied Software Research (drh@hwaci.com). Sqlite dapat di-download di <http://www.sqlite.org>. Namun, sebelum men-download, cobalah cari terlebih

dahulu ke dalam CD-ROM Anda. Hampir semua distro desktop umumnya telah meletakkan sqlite ke dalam CD/DVD distro.

Saat ini, SQLite telah mencapai versi 3.x, walaupun versi 2.x masih tetap dimaintain. Karena versi 2.x masih sangat banyak digunakan, maka pada distribusi versi 3, beberapa penyesuaian pun dilakukan. Sebagai contoh, nama header pustaka pada versi 3 adalah `sqlite3.h`, dan bukan `sqlite.h`. Dengan beberapa penyesuaian tersebut, maka dimungkinkan adanya dua versi SQLite terinstall dalam sistem.

Berikut ini kita akan membahas beberapa fitur baru versi 3, sekaligus membandingkan dengan versi 2.

Format file baru

Format file data SQLite telah mengalami perubahan dibandingkan dengan versi 2. Sayangnya, perubahan format tersebut tidak kompatibel satu sama lainnya. Versi 2 tidak akan membaca database versi 3, dan sebaliknya.

Untungnya, walaupun format data tidak kompatibel, migrasi tetap dapat dilakukan dengan sangat mudah dengan memanfaatkan perintah dump dari program `sqlite` versi 2, dan hasilnya kemudian disimpan pada database versi 3 menggunakan program

`sqlite` versi 3. Hal ini dimungkinkan karena dalam satu sistem bisa terinstal lebih dari satu versi SQLite.

Secara teknis, format file baru SQLite menggunakan tipe data B+Tree untuk tabel (untuk indeks, b-tree konvensional masih dipergunakan). Pada tree ini, semua data disimpan pada daun-daun tree. Menurut pengembang SQLite, format baru ini akan memungkinkan skalabilitas yang lebih baik.

Tipe data

Dukungan untuk BLOB dan tipe data. Selama ini, SQLite dikenal sebagai database server yang tidak bertipe (walaupun pada DDL didukung). Pada SQLite 2.x, semua data akan dikonversi ke teks ASCII. Pada versi 3, apabila memungkinkan, maka data akan disimpan sebagai tipe yang didefinisikan oleh user. Penyimpanan ke format non-ASCII sangat penting untuk memungkinkan adanya tipe data BLOB (Binary Large Object). BLOB pada versi 2.x dimungkinkan apabila data telah diencode terlebih dahulu.

Masalah tipe data ini merupakan hal yang sangat penting. Pada versi 2.8, karena semua data akan dikonversi ke teks ASCII, maka dengan demikian, data yang seharusnya disimpan ke kolom satu bisa

SQLite home page - Mozilla Firefox

SQLite home page

SQLite to WEB: testdb

About SQLite

SQLite is a small C library that implements a self-contained, embeddable, zero-configuration SQL database engine. Features include:

- Transactions are atomic, consistent, isolated, and durable (ACID) even after system crashes and power failures.
- Zero-configuration - no setup or administration needed.
- Implements most of SQL92. ([Features not supported](#))
- A complete database is stored in a single disk file.
- Database files can be freely shared between machines with different byte orders.
- Supports databases up to 2 terabytes (2^{41} bytes) in size.
- Sizes of strings and BLOBs limited only by available memory.
- Small code footprint: less than 30K lines of C code, less than 250KB code space (gcc on i486)
- **Faster** than popular client/server database engines for most common operations.
- Simple, easy to use [API](#).
- [TCL bindings](#) included. Bindings for many other languages [available separately](#).
- Well-commented source code with over 95% test coverage.
- Self-contained: no external dependencies.
- Sources are in the [public domain](#). Use for any purpose.

The SQLite distribution comes with a standalone command-line access program (`sqlite`) that can be used to administer an SQLite database.

News

2005-Feb-19 - Version 3.1.3 Released

Version 3.1.3 cleans up some minor issues discovered in version 3.1.2.

2005-Feb-15 - Versions 2.8.16 and 3.1.2 Released

A critical bug in the `VACUUM` command that can lead to database corruption has been fixed in both the 2.x branch and the main 3.x line. This bug has existed in all prior versions of SQLite. Even though it is unlikely you will ever encounter this bug, it is suggested that all users upgrade. See [ticket #1116](#) for additional information.

Version 3.1.2 is also the first stable release of the 3.1 series. SQLite 3.1 features added support for correlated subqueries, autovacuum, `autoincrement`, `ALTER TABLE`, and other enhancements. See the [release notes for version 3.1.0](#) for a detailed description of the changes available in the 3.1 series.

2004-Nov-09 - SQLite at the 2004 International PHP Conference

There was a talk on the architecture of SQLite and how to optimize

Done

Website SQLite.

disimpan ke kolom lain. Hal ini mungkin terdengar sangat asing pada database system lain. Pada SQLite versi 3, fitur ini masih dipertahankan, namun penanganannya telah diperbaiki. Pada saat data dimasukkan ke dalam kolom, SQLite 3 akan mencoba untuk melakukan konversi ke format yang telah didefinisikan. Apabila konversi tidak dimungkinkan, SQLite 3 tetap akan menyimpan data tersebut.

Sebagai contoh, apabila data string ingin dimasukkan ke dalam kolom integer, maka SQLite akan memeriksa apakah data dapat dikonversi ke bilangan. Apabila dapat, maka konversi dilakukan dan data akan disimpan ke dalam kolom integer. Apabila tidak dapat dikonversi, maka data akan tetap disimpan sebagai string.

Berikut ini adalah tipe-tipe data yang didukung:

- NULL, nilai null
- INTEGER, menyimpan integer sampai integer 8 byte
- REAL, menyimpan nilai floating point/pecahan sampai 8 byte IEEE floating point
- TEXT, menyimpan teks
- BLOB, menyimpan data blob

Concurrency

Sebuah database tentunya bisa diakses oleh lebih dari satu orang pada saat yang sama. Dan, dengan kondisi demikian, maka locking adalah isu yang sangat penting. Pada SQLite 3, locking pada level tabel juga telah diimplementasikan, walaupun tidak akan sebaik *locking* pada database system besar lain.

ROWID 64bit

Setiap baris dalam tabel akan memiliki ID yang unik (rowid). Dibandingkan dengan versi 2.8 yang hanya mendukung rowid sampai 32bit, versi 3 telah mendukung sampai 64bit.

Dukungan UTF-8 dan UTF-16

SQLite 3 telah mendukung baik teks UTF-8 dan UTF-16 secara natif. Hal ini menjadikan dukungan yang lebih luas untuk bahasa di dunia. Implementasi saat ini belum sepenuhnya mendukung kedua encoding tersebut, namun, di masa depan, SQLite 3 akan mendukung sepenuhnya.

Fitur SQL92 yang tidak didukung

SQLite berusaha untuk mendukung sebanyak mungkin fitur SQL92. Namun, untuk kecepatan dan alasan lain, beberapa fitur tidak didukung. Berikut ini adalah fitur SQL92 yang tidak didukung:

- CHECK constrain. Parser mengenali CHECK namun tidak akan dijalankan, kecuali UNIQUE dan NOT NULL.
- FOREIGN KEY. Parser mengenali namun tidak akan menjalankan.
- Dukungan trigger komplit. Saat ini, dukungan trigger pada SQLite belum komplit.
- ALTER TABLE. Untuk mengubah tabel, hapuslah tabel dan buatlah kembali. Saat ini, hanya ALTER TABLE ... RENAME TABLE ... yang didukung.
- Transaksi bersarang (nested transaction).
- Fungsi COUNT(DISTINCT X). Cara alternatif adalah dengan menggunakan subquery seperti SELECT count(x) FROM (SELECT DISTINCT x FROM tbl);
- RIGHT OUTER JOIN dan FULL OUTER JOIN tidak diimplementasikan. Yang telah diimplementasikan adalah LEFT OUTER JOIN.
- Menulis ke view tidak diimplementasikan. Pada SQLite, view adalah read-only.
- GRANT dan REVOKE. Hal ini disebabkan karena pengguna file database yang disimpan langsung pada filesystem, sehingga hak akses akan menuruti hak akses pada filesystem.

Bisa dilihat bahwa fitur yang tidak didukung sudah cukup sedikit dan tidak terlalu vital. Yang mungkin sedikit vital adalah FOREIGN KEY, yang terpaksa harus dicapai melalui program.

Distribusi SQLite

Distribusi SQLite 3 sebenarnya hanya terdiri dari satu program dan beberapa pustaka C untuk mengakses database. Program `sqlite3` yang datang bersama distribusi SQLite3 merupakan program untuk bekerja dengan SQLite, mulai dari pembuatan database sampai memungkinkan perintah-perintah khusus SQLite dan sintaks SQL untuk dijalankan.

Secara sederhana, program `sqlite3` bisa dianggap seperti halnya program `psql` (client PostgreSQL) dan `mysql` (client MySQL).

Sementara, pustaka yang datang bersama SQLite adalah pustaka `libsqlite3`. Pustaka ini dapat digunakan oleh developer untuk bekerja dengan database SQLite.

Umumnya, paket-paket dalam distro dibagi dalam paket `sqlite` dan `sqlite-devel`. Untuk pengembangan program, paket `sqlite-devel` harus diinstal. Paket `sqlite-devel` akan berisi dokumentasi, file-file header dan pustaka untuk kebutuhan pembuatan program.

Menggunakan `sqlite3`

Begitu dijalankan, `sqlite3` dapat menerima dua macam perintah: perintah `sqlite3` ataupun perintah SQL. Perintah `sqlite3` adalah perintah yang diawali oleh tanda titik dan dapat menerima parameter. Berikut ini adalah beberapa perintah `sqlite3` yang berguna:

- `.help` berguna untuk melihat daftar perintah `sqlite3`.
- `.databases` berguna untuk melihat nama dan file yang digunakan database.
- `.dump` berguna untuk melakukan dump database.
- `.echo` berguna untuk menampilkan perintah yang dijalankan dalam hasil.
- `.mode` berguna untuk mengatur mode output. Mode yang tersedia cukup banyak, diantaranya `line`, `html`, `column`, `csv` dan lain sebagainya.
- `.output` berguna untuk menyimpan output perintah ke file tertentu.
- `.read` berguna untuk menjalankan SQL dari file tertentu.
- `.tables` berguna untuk menampilkan daftar tabel.

Untuk perintah SQL, Anda dapat langsung memberikannya di prompt.

Membuat database baru

Untuk membuat database baru, cukup jalankan program `sqlite3`, diikuti oleh sebuah parameter berupa nama database. Ekstensi nama file database umumnya adalah `.db`, namun Anda dapat menggunakan ekstensi yang Anda suka. Tidak ada pembatasan.

Setelah itu, masukkanlah perintah SQL untuk membuat tabel ataupun mengisi data ke dalam tabel. Setelah itu, begitu Anda keluar dari `sqlite3`, database telah tersimpan.

Contoh:

```
$ sqlite3 test.db
SQLite version 3.1.3
Enter ".help" for instructions
sqlite> create table test(a
integer, b integer);
sqlite> .exit
```

Melihat daftar table

Untuk melihat daftar table, berikanlah perintah .tables seperti contoh berikut ini:

```
$ sqlite3 test.db
SQLite version 3.1.3
Enter ".help" for instructions
sqlite> .tables
test test2 test3 test4
test5
sqlite> .exit
```

Menjalankan sintaks SQL dari command line

Ada kalanya, Anda ingin menjalankan sintaks SQL dari command line, misalnya ketika digunakan dalam script. Untuk kebutuhan tersebut, Anda bisa langsung memberikan perintah SQL langsung pada command line seperti contoh berikut:

```
$ sqlite3 test.db "select * from test";
1|1
2|2
```

Sebenarnya, Anda pun bisa saja menjalankan perintah internal sqlite3 di command line:

```
$ sqlite3 test.db ".tables";
test test2 test3 test4
test5
```

Menjalankan sintaks SQL dari file lain

Untuk menjalankan sintaks SQL yang tersimpan dalam file lain, gunakanlah perintah internal .read seperti contoh berikut ini.

isi file test.sql

```
select * from test;
insert into test(a,b)
values(3,3);
select * from test;
```

Contoh penggunaan .read:

```
$ sqlite3 test.db
SQLite version 3.1.3
```

Enter ".help" for instructions

```
sqlite> .echo on
sqlite> .read test.sql
.read test.sql
select * from test;
1|1
2|2
3|3
insert into test(a,b)
values(3,3);
select * from test;
1|1
2|2
3|3
3|3
sqlite> .exit
```

Dump database ke file

Walaupun back-up dapat dilakukan dengan langsung meng-copy file database ke lokasi lain, tidak harus dengan cara dump seperti MySQL ataupun PostgreSQL, ada kalanya dump diperlukan. Contoh paling baik adalah ketika Anda ingin memigrasi data dari SQLite versi lama ke SQLite versi baru.

Berikut ini adalah langkah-langkah melakukan dump:

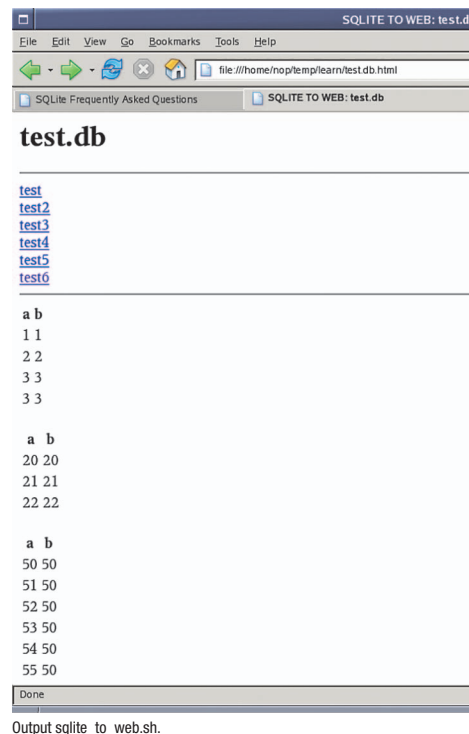
- jalankan sqlite3
- berikan perintah .output <NAMA_FILE_HASIL_DUMP>
- berikan perintah .dump
- atur kembali agar output kembali disimpan ke stdout dengan perintah .output stdout

Contoh:

```
$ sqlite3 test.db
SQLite version 3.1.3
Enter ".help" for instructions
sqlite> .output dump.sql
sqlite> .dump
sqlite> .output stdout
sqlite> .exit
```

Contoh file hasil dump:

```
$ cat dump.sql
BEGIN TRANSACTION;
CREATE TABLE test(a integer, b integer);
INSERT INTO "test" VALUES(1, 1);
INSERT INTO "test" VALUES(2, 2);
INSERT INTO "test" VALUES(3, 3);
```



```
INSERT INTO "test" VALUES(3, 3);
CREATE TABLE test2(a integer, b integer);
CREATE TABLE test3(a integer, b integer);
CREATE TABLE test4(a integer, b integer);
CREATE TABLE test5(a integer, b integer);
COMMIT;
```

Melihat skema database

Untuk melihat skema dalam database, berikanlah perintah .schema seperti contoh berikut:

```
$ sqlite3 test.db
SQLite version 3.1.3
Enter ".help" for instructions
sqlite> .schema
CREATE TABLE test(a integer, b integer);
CREATE TABLE test2(a integer, b integer);
CREATE TABLE test3(a integer, b integer);
CREATE TABLE test4(a integer, b integer);
CREATE TABLE test5(a integer, b integer);
sqlite> .exit
```

Untuk melihat skema per table, berikan parameter nama tabel pada .schema:

```
sqlite> .schema test
CREATE TABLE test(a integer, b
integer);
sqlite>
```

Membuat auto-increment

SQLite tidak memiliki dukungan khusus untuk membuat field auto-increment. Apa yang bisa dilakukan untuk membuat field auto-increment adalah dengan membuat suatu field bertipe integer dan menjadikannya primary key. Pada saat insert, pengguna tidak perlu memasukkan nilai tertentu ke field tersebut. SQLite akan menambahkan sendiri, sehingga auto-increment pun dimungkinkan.

Sebagai contoh, kita akan membuat table test6 yang terdiri dari dua field berikut ini:

- a bertipe integer dan berupa primary key
- b bertipe integer

Berikut ini adalah sintaks SQL untuk kebutuhan pembuatan tabel:

```
sqlite> create table test6(a
integer primary key, b integer);
```

Kita telah mempersiapkan field a sebagai field auto-increment. Dengan tidak memasukkan nilai ke field a setiap kali melakukan insert, nilai data pada field a akan ditambahkan satu. Contoh:

```
sqlite> insert into test6(b)
values(1);
sqlite> insert into test6(b)
values(2);
sqlite> insert into test6(b)
values(3);
```

Berikut ini adalah isi table test6:

```
select * from test6;
1|1
2|2
3|3
```

Proses penambahan ini akan berhenti apabila telah mencapai nilai 2147483647. Selanjutnya, nilai random yang akan dimasukkan.

Mengubah prompt sqlite3

Secara default, prompt pada sqlite3 adalah sqlite> dan prompt untuk baris lanjutan

perintah yang belum selesai adalah ...>. Anda bisa menggantinya dengan perintah .prompt apabila diinginkan.

Contoh:

```
$ sqlite3 test.db
SQLite version 3.1.3
Enter ".help" for instructions
sqlite> select
...> *
...> from
...> test;
1|1
2|2
3|3
3|3
sqlite> .prompt mydb> ' --->'
mydb>select
--->*
--->from
--->test;
1|1
2|2
3|3
3|3
mydb>
```

Output ke format CSV

Anda bisa membuka isi dari suatu tabel database sqlite di OpenOffice.org ataupun di Excel. Caranya adalah dengan membuat output program sqlite3 menjadi format CSV. Setiap record kemudian akan ditampilkan perbaris, dan setiap field akan ditampilkan dengan dipisahkan oleh koma. Berikut ini contohnya:

```
$ sqlite3 test.db
SQLite version 3.1.3
Enter ".help" for instructions
sqlite> .mode csv
sqlite> .output test.csv
sqlite> select * from test;
sqlite> .output stdout
sqlite> .mode list
sqlite> .exit
```

Berikut ini adalah isi file test.csv:

```
1,1
2,2
3,3
3,3
```

Output ke format HTML

Apabila isi table ingin ditampilkan di inter-

net, maka format output bisa diganti ke format HTML. Berikut ini adalah contohnya:

```
$ sqlite3 test.db
SQLite version 3.1.3
Enter ".help" for instructions
sqlite> .header on
sqlite> .mode html
sqlite> .output test.html
sqlite> select * from test;
sqlite> .output stdout;
sqlite> .mode list
sqlite> .header off
sqlite> .exit
```

Berikut ini adalah isi test.html:

```
<TR><TH>a</TH><TH>b</TH></TR>
<TR><TD>1</TD>
<TD>1</TD>
</TR>
<TR><TD>2</TD>
<TD>2</TD>
</TR>
<TR><TD>3</TD>
<TD>3</TD>
</TR>
<TR><TD>3</TD>
<TD>3</TD>
</TR>
```

Sebagai catatan, perintah .header on berguna untuk meminta sqlite3 untuk menampilkan nama field setiap kali query dilakukan. Tentunya, kita tidak ingin tabel HTML yang dibuat tidak memiliki judul, bukan?

Output teks yang rapi

Output pada sqlite3 memang tidak rapi. Apabila Anda menginginkannya untuk tampil rapi seperti psql misalnya, gantilah format output menjadi column, dan jangan lupa untuk mengatur lebar kolom data Anda. Berikut ini adalah contohnya:

```
$ sqlite3 test.db
SQLite version 3.1.3
Enter ".help" for instructions
sqlite> .header on
sqlite> .mode column
sqlite> .width 20 20
sqlite> select * from test;
a                b
-----
1                1
2                2
```

```
3          3
3          3
sqlite> .exit
```

Sebagai catatan, perintah `.width` dan parameter `20 20` dimaksudkan untuk mengubah lebar kolom pertama sebesar 20 karakter dan lebar kolom kedua sebesar 20 karakter. Dengan cara seperti ini, lebar kolom tertentu bisa diatur dengan fleksibel.

Vakum database

Pada saat Anda menghapus banyak record dari database, ukuran file database umumnya tidak akan berkurang. Hal ini wajar karena ruang kosong (yang tadinya digunakan untuk menampung data sebelum dihapus) akan ditandai sebagai ruang kosong internal. Ruang kosong internal tersebut nantinya akan digunakan begitu ada data baru dimasukkan.

Apabila Anda tidak senang dengan cara kerja demikian dan ingin mendapatkan pengurangan ukuran database begitu data dihapus (SQLite sebelum 3.1, karena pada 3.1, `auto vacuum` menjadi default), Anda bisa menggunakan perintah `VACUUM`. Perintah yang satu ini akan membangun ulang database dari awal. Hal ini akan sedikit memakan waktu dan space harddisk. Pada sistem dimana SQLite dibangun, vakum akan memakan waktu setengah detik untuk setiap MB data, dan akan memakan space dua kali ukuran database.

Berikut ini adalah contoh penggunaan perintah `vacuum`:

```
$ sqlite3 test.db
SQLite version 3.1.3
Enter ".help" for instructions
sqlite> vacuum;
sqlite> .exit
```

Contoh aplikasi : iterasi tabel dan menampilkan hasilnya di web

Pada contoh sebelumnya, kita telah melihat bagaimana mengubah format output dari perintah SQL yang dijalankan ke HTML. Ini berarti, kita dapat membukanya di web browser atau menampilkannya di Internet.

Namun, bagaimana kalau kita ingin menampilkan semua tabel dan isinya ke internet dengan cara yang mudah? Untuk lebih mudahnya, kita bisa membangun shell script sederhana yang akan membuat se-

buah HTML tunggal untuk semua tabel dan isinya. Berikut ini adalah source code shell script tersebut:

```
#!/bin/sh

# (c) Noprianto
# 20 June 2005

SQLITE=/usr/bin/sqlite3
OUT=$1.html

HEADER="<HTML><HEAD><TITLE>
SQLite TO WEB: $1</TITLE></
HEAD><BODY><H1>$1</H1><HR
NOSHADE>"
FOOTER="<HR NOSHADE><I>Generated
by `basename $0` on `date`</I></
BODY></HTML>"
LINKS=""
TABLE_CONTENTS=""

echo "sqlite_to_web.sh"
echo "(c) Nop, 2005"

#checking routine
[ ! -x $SQLITE ] && echo "Sorry:
SQLite binary not found" &&
exit 1
[ -z $1 ] && echo "Sorry: I need
one parameter, which is your
database" && exit 2
[ ! -e $1 ] && echo "Sorry:
Database file you specified
cannot be found" && exit 3
[ ! -r $1 ] && echo "Sorry:
Database file you specified
cannot be read" && exit 4

#get table list
TABLES=`$SQLITE $1 .tables`

#create table HTML code
for i in $TABLES
do
    LINK="<A HREF=\"$#i\">$i
</A><BR>"
    TABLE_CONTENT="<A NAME=\
\"$i\"></A><TABLE>`$SQLITE
-header -html $1 `select *
from $i`;`</TABLE><BR>"
    LINKS="$LINKS $LINK"
```

```
TABLE_CONTENTS="$TABLE_
CONTENTS $TABLE_CONTENT"
done

echo "<HEADER $LINKS <HR
NOSHADE> $TABLE_CONTENTS
$FOOTER" > $OUT

echo DONE.
```

Catatan program

- Untuk secara otomatis menampilkan header, berikanlah parameter `-header` pada program `sqlite3`
- Dan, untuk secara mengotomatis mengubah format output ke HTML, berikan parameter `-html`.

Cara kerja program:

- Pertama-tama, script akan memeriksa apakah binary `SQLite` tersedia atau tidak. Pengguna dapat mengubah lokasi pencarian dengan mengubah path ke binary `SQLite` pada variabel `$SQLITE`.
- Setelah itu, pemeriksaan akan dilanjutkan kepada pemeriksaan parameter pertama (apakah pengguna memasukkan parameter pertama atau tidak).
- Kemudian, dilanjutkan dengan memeriksa ketersediaan database di filesystem (apakah file ditemukan).
- Dan pemeriksaan diakhiri dengan memeriksa apakah database dapat dibaca atau tidak (pemeriksaan filesystem semata, karena database `sqlite` disimpan sebagai file biasa).
- Daftar table pun kemudian didapat dengan memanggil binary `SQLite` diikuti dengan perintah `.tables` (merupakan perintah terpisah).
- Berdasarkan daftar table yang didapat, kita pun membuat link dan table isi table database.
- Akhirnya, semua variabel yang didapatkan kita tuliskan di HTML output.

Selanjutnya, script tersebut bisa digunakan untuk menampilkan keseluruhan isi semua table di dalam database `SQLite` Anda. Dan, dengan script tersebut pulalah, kita menutup artikel ini. Sampai di sini dulu perkenalan kita dengan database `SQLite`. Selamat mencoba! 🙏

Noprianto (noprianto@infolinux.co.id)