Mengenal SQLite *Command Line*

Didik Setiawan
di2k.setiawan@gmail.com

Pendahuluan

SQLite adalah *Relational Database Management Server* (RDBMS) alternatif yang bersifat *portable* (tidak memerlukan proses instalasi), cepat, gratis, dan didukung oleh banyak bahasa pemrograman.

Keunggulan SQLite antara lain :

1. *Portable* tidak perlu proses instalasi, cukup menggunakan satu file `sqlite3.exe`;
2. *Flat file* (satu database satu file) ;
3. Mendukung *transaction* dan *view*;
4. Sangat cepat, karena berupa *flat file*;
5. Menggunakan *Query Language* yang mirip dengan RDBMS pada umumnya.

Salah satu *library* yang terdapat dalam database SQLite adalah fungsi *command-line sqlite3* (`sqlite3.exe`) yang memungkinkan user untuk berinteraksi dan menjalankan perintah SQL di SQLite.

Tulisan ini berisi pengenalan database SQLite dan perintah (*Command Line*) yang dapat digunakan pada database SQLite. Sebagian besar contoh yang digunakan dalam tulisan ini menggunakan SQLite 3 (`sqlite3`) pada sistem operasi Windows.

Instalasi

SQLite bersifat *portable* sehingga tidak dibutuhkan proses instalasi, untuk menggunakan SQLite dapat dilakukan dengan cara : download `sqlite3.zip` (untuk windows) pada alamat : http://www.sqlite.org, lalu ekstrak file yang telah didownload tersebut hingga menghasilkan file `sqlite3.exe`, selanjutnya SQLite dapat diakses melalui *command prompt*.

Untuk menjalankan perintah `sqlite3`, ketikkan `sqlite3` diikuti dengan nama file yang juga merupakan nama database. Jika file dimaksud belum ada, otomatis akan dibuatkan sebuah file baru. Untuk menjalankan perintah SQL dilakukan dengan menambahkan tanda *semicolon* (;), lalu tekan enter dan untuk mengeksekusi perintah SQL.

Contoh : Untuk membuat database baru dengan nama `db_anggaran` yang berisi tabel `t_dept` dapat dilakukan dengan perintah sebagai berikut :

```
D:\sqlite>sqlite3 db_anggaran
SQLite version 3.6.11
Enter ".help" for instructions
Enter SQL statements terminated with a ";"
sqlite> create table t_dept(kddept char(3), nmdept varchar(200));
```

Untuk mengakhiri proses jalannya perintah SQL dapat dilakukan dengan menggunakan *interrupt character* (`Ctrl+C`). Pastikan bahwa setiap akhir perintah SQL di sertai tanda *semicolon* (`;`). `Sqlite3` melihat *semicolon* sebagai tanda bahwa perintah SQL yang diberikan telah lengkap. `Sqlite3` akan menganggap perintah tanpa *semicolon* (`;`) sebagai satu kesatuan perintah dengan perintah sebelumnya. Ketentuan ini memungkinkan untuk membuat perintah SQL lebih dari satu baris perintah.

SQLite *Dot Command*

Terdapat beberapa perintah khusus yang hanya digunakan di `sqlite3`. Perintah khusus ini ditandai dengan tanda titik/*dot command* (.), *dot command* merupakan perintah yang hanya akan diterjemahkan oleh `sqlite3` sebagai perintah internal.

Perintah *dot command* biasanya digunakan untuk melakukan perubahan pada format output hasil *query* atau untuk menjalankan perintah *query* yang bersifat *prepackaged*.

Daftar perintah yang termasuk ke dalam `sqlite3` *dot command* dapat dilihat dengan perintah berikut :

```
sqlite> .help
.backup ?DB? FILE          Backup DB (default "main") to FILE
.bail ON|OFF               Stop after hitting an error. Default OFF
.databases                 List names and files of attached databases
.dump ?TABLE? ...          Dump the database in an SQL text format
                        If TABLE specified, only dump tables matching
                          LIKE pattern TABLE.
.echo ON|OFF               Turn command echo on or off
.exit                      Exit this program
.explain ?ON|OFF?          Turn output mode suitable for EXPLAIN on or
off.
                        With no args, it turns EXPLAIN on.
.genfkey ?OPTIONS?         Options are:
                            --no-drop: Do not drop old fkey triggers.
                            --ignore-errors: Ignore tables with fkey
errors
                            --exec: Execute generated SQL immediately
                        See file tool/genfkey.README in the source
                          distribution for further information.
.header(s) ON|OFF          Turn display of headers on or off
.help                      Show this message
.import FILE TABLE         Import data from FILE into TABLE
.indices ?TABLE?           Show names of all indices
                        If TABLE specified, only show indices for tables
                        matching LIKE pattern TABLE.
.load FILE ?ENTRY?         Load an extension library
.mode MODE ?TABLE?         Set output mode where MODE is one of:
                        csv        Comma-separated values
                        column     Left-aligned columns. (See .width)
                        html       HTML <table> code
                        insert SQL insert statements for TABLE
                        line       One value per line
                        list       Values delimited by .separator string
                        tabs       Tab-separated values
                        tcl        TCL list elements
.nullvalue STRING          Print STRING in place of NULL values
.output FILENAME           Send output to FILENAME
.output stdout             Send output to the screen
.prompt MAIN CONTINUE      Replace the standard prompts
.quit                      Exit this program
.read FILENAME             Execute SQL in FILENAME
.restore ?DB? FILE         Restore content of DB (default "main") from
FILE
.schema ?TABLE?            Show the CREATE statements
                        If TABLE specified, only show tables matching
                        LIKE pattern TABLE.
.separator STRING          Change separator used by output mode and
.import
.show                      Show the current values for various settings
.tables ?TABLE?            List names of tables
                        If TABLE specified, only list tables matching
                        LIKE pattern TABLE.
.timeout MS                Try opening locked tables for MS milliseconds
.width NUM1 NUM2 ...       Set column widths for "column" mode
.timer ON|OFF              Turn the CPU timer measurement on or off
```

1. Format *Query*

`sqlite3` dapat menampilkan hasil *query* dalam delapan format yang berbeda: "`csv`", "`column`", "`html`", "`insert`", "`line`", "`list`", "`tabs`", dan "`tcl`". Berikut beberapa perintah yang dapat digunakan untuk mengatur tampilan/format hasil *query* :

1) `.mode`

Perubahan format dapat dilakukan dengan perintah `.mode`, secara default format yang digunakan adalah `list`. Dalam format `list` seluruh record hasil *query* ditulis dalam satu baris output untuk setiap kolom dengan pemisahan menggunakan separator *string pipe symbol* (|), contoh :

```
sqlite> .mode list
sqlite> select * from t_dept;
001|MAJELIS PERMUSYAWARATAN RAKYAT
002|DEWAN PERWAKILAN RAKYAT
sqlite>
```

2) `.sparator`

Perintah `. sparator` *dot command* dapat digunakan untuk melakukan perubahan karakter pemisah pada *list mode*. Contoh pemisahan menggunakan koma dan spasi (‘) dapat dilakukan dengan cara sebagai berikut :

```
sqlite> .separator ", "
sqlite> select * from t_dept;
001, MAJELIS PERMUSYAWARATAN RAKYAT
002, DEWAN PERWAKILAN RAKYAT
sqlite>
```

3) `.line`

Dalam format `line`, setiap kolom data ditampilkan sebagai sebuah baris, setiap baris terdiri atas nama kolom, tanda sama dengan dan kolom data. Pembeda *record* ditandai dengan baris kosong, contoh :

```
sqlite> .mode line
sqlite> select * from t_dept;
kddept = 001
nmdept = MAJELIS PERMUSYAWARATAN RAKYAT

kddept = 002
nmdept = DEWAN PERWAKILAN RAKYAT
sqlite>
```

Dalam format `column`, setiap record akan ditampilkan sebagai berikut :

```
sqlite> .mode column
sqlite> select * from t_dept;
kddept  nmdept
------  ------------------------------
001     MAJELIS PERMUSYAWARATAN RAKYAT
002     DEWAN PERWAKILAN RAKYAT
sqlite>
```

4) `.width`

Perintah `. width` digunakan untuk mengatur lebar kolom, dapat digunakan jika lebar kolom yang ada tidak dapat menampilkan seluruh isi data. Jika lebar kolom didefinisikan dengan 0, maka lebar kolom akan otomatis menyesuaikan dengan lebar pada data pertama. Default seting untuk setiap kolom adalah *auto-adjusting* bernilai 0.

5) `.header`

Label atau nama kolom dapat ditampilan atau tidak dengan menggunakan perintah `. header`, berikut contoh *header* bernilai off :

```
sqlite> .header off
sqlite> select * from t_dept;
001   MAJELIS PERMUSYAWARATAN RAKYAT
002   DEWAN PERWAKILAN RAKYAT
sqlite>
```

6) `.insert`

Perintah `. insert` digunakan untuk memformat hasil *query* tampak seperti perintah SQL `INSERT`. Format ini dapat digunakan untuk menghasilkan text yang dapat digunakan untuk memindahkan data ke database lain.

Dalam perintah `.insert` dapat ditambahkan argument yang menunjukkan nama tabel yang akan menjadi tujuan insert, contoh :

```
sqlite> .mode insert new_dept
sqlite> select * from t_dept;
INSERT INTO 'new_dept' VALUES('001', 'MAJELIS PERMUSYAWARATAN RAKYAT');
INSERT INTO 'new_dept' VALUES('002', 'DEWAN PERWAKILAN RAKYAT');
sqlite>
```

7) `.html`

Perintah `.html` akan menghasilkan *query* sebagai tabel XHTM.

Hasil *query* diawali dengan <TABLE> namun penutup </TABLE> tidak disertakan, seluruh format <TR>, <TH>, dan <TD> disertakan untuk menjaga kesesuaian dengan format CGI.

2. Menulis Hasil *Query* ke File

Secara default, sqlite3 menampilkan hasil *query* ke layar. Untuk memindahkan hasil *query* ke dalam file dapat dilakukan dengan menggunakan perintah .output sebagaimana contoh berikut :

```
sqlite> .mode list
sqlite> .separator |
sqlite> .output test_file_1.txt
sqlite> select * from t_dept;
sqlite> .exit
```

3. Skema Database

Skema database SQLite disimpan pada sebuah tabel khusus yang bernama sqlite_master. Skema database pada tabel sqlite_master dapat diakses dengan menggunakan perintah SELECT, contoh :

```
$ sqlite3 db_anggaran
SQLite vresion 3.6.11
Enter ".help" for instructions
sqlite> select * from sqlite_master;
type = table
name = t_dept
tbl_name = t_dept
rootpage = 3
        sql = create table t_dept(kddept char(3), nmdept varchar(200))
sqlite>
```

Perintah DROP TABLE, UPDATE, INSERT atau DELETE tidak dapat digunakan pada tabel sqlite_master. Tabel sqlite_master akan otomatis ter update saat terjadi penambahan atau perubahan tabel di database.

Sqlite3 menyediakan beberapa perintah *dot command* yang dapat digunakan untuk melihat skema database, yaitu :

1) .tables

Perintah .tables digunakan untuk menampilkan daftar tabel yang terdapat dalam sebuah database.

```
sqlite> .tables
t_dept
sqlite>
```

Perintah `.tables` merupakan *shortcut* untuk baris perintah berikut :

```
SELECT name FROM sqlite_master
WHERE type IN ('table','view') AND name NOT LIKE 'sqlite_%'
UNION ALL
SELECT name FROM sqlite_temp_master WHERE type IN ('table','view')
ORDER BY 1
```

2) `.schema`

Perintah `.schema` tanpa parameter digunakan untuk menampilkan perintah `CREATE TABLE` dan `CREATE INDEX` yang digunakan untuk membuat seluruh tabel database, jika nama tabel disertakan setelah perintah `.schema` akan menampilkan perintah `CREATE` yang digunakan untuk membuat tabel tersebut, contoh:

```
sqlite> .schema t_dept
CREATE TABLE t_dept(kddept char(3), nmdept varchar(200));
sqlite>
```

Perintah `.schema` serupa dengan perintah *list mode* yang kemudian diikuti perintah berikut :

```
SELECT sql FROM
(SELECT * FROM sqlite_master UNION ALL SELECT * FROM sqlite_temp_master)
WHERE type!='meta'
ORDER BY tbl_name, type DESC, name
```

Atau perintah berikut untuk `.schema` yang disertai nama tabel :

```
SELECT sql FROM
(SELECT * FROM sqlite_master UNION ALL SELECT * FROM sqlite_temp_master)
WHERE tbl_name LIKE '%s' AND type!='meta' AND sql NOT NULL AND name NOT
LIKE 'sqlite_%'
ORDER BY substr(type,2,1), name
```

Tanda `%s` akan diisi nilainya dengan *argument* yang diberikan.

3) `.databases`

Perintah `.databases` akan menampilkan seluruh databases yang sedang terbuka, terdiri sekurangnya dua database, pertama '`main`' untuk database yang terbuka dan '`temp`' sebagai database yang digunakan untuk menampung *temporary tabel*.

```
sqlite> .databases
seq name             file
--- --------------- ----------------------------
```

```
0   main            D:\sqlite\db_anggaran
1   temp
sqlite>
```

4) .dump

Perintah .dump digunakan untuk melakukan konversi isi database ke dalam satu file text ASCII. File ASCII hasil bentukan .dump dapat di konversi kembali menjadi database SQLite. Berikut contoh dalam perintah Linux :

```
$ echo '.dump' | sqlite3 db_anggaran | gzip -c
>db_anggaran.dump.gz
```

Hasil *generate* berupa file dengan nama db_anggaran.dump.gz yang berisi segala hal yang dibutuhkan untuk melakukan rekonstruksi database. untuk rekonstruksi database dapat dilakukan dengan perintah :

```
$ zcat db_anggaran.dump.gz | sqlite3 db_anggaran2
```

Format file hasil perintah .dump adalah dalam bentuk SQL sehingga dapat digunakan untuk melakukan import data ke database lain.

5) .explain

Perintah .explain merupakan perintah spesifik SQLite yang berguna untuk melakukan *debugging*. Jika perintah SQL diawali dengan EXPLAIN, maka SQL akan melakukan parsing dan menganalisa proses yang ada, contoh :

```
sqlite> .explain
sqlite> explain delete from t_dept where kddept<'002';
addr opcode         p1   p2   p3   p4             p5 comment
---- -------------- ---- ---- ---- -------------- -- -------------
0    Trace          0    0    0                   00
1    Goto           0    19   0                   00
2    Null           0    1    0                   00
3    String8        0    3    0    002            00
4    OpenRead       0    2    0    1              00
5    Rewind         0    11   0                   00
6    Column         0    0    4                   00
7    Ge             3    10   4    collseq(BINARY) 69
8    Rowid          0    2    0                   00
9    RowSetAdd      1    2    0                   00
10   Next           0    6    0                   01
11   Close          0    0    0                   00
12   OpenWrite      0    2    0    2              00
13   RowSetRead     1    17   2                   00
14   NotExists      0    16   2                   00
15   Delete         0    1    0    t_dept         00
16   Goto           0    13   0                   00
17   Close          0    0    0                   00
```

```
18    Halt          0    0    0                    00
19    Transaction   0    1    0                    00
20    VerifyCookie  0    1    0                    00
21    TableLock     0    2    1    t_dept          00
22    Goto          0    2    0                    00
```

6) `.timeout`

Perintah `.timeout` digunakan untuk menentukan waktu berapa lama `sqlite3` akan melakukan penguncian (*lock*) untuk membersihkan file yang akan diakses sebelum mengembalikan nilai kesalahan.

Default nilai *timeout* adalah nol (0) sehingga kesalahan akan segera dimunculkan jika tabel atau index sedang di *lock*.

Perintah `.exit` digunakan untuk mengakhiri `sqlite3`.

Perintah SQL dalam SQLite

Berikut adalah beberapa perintah SQL yang dapat digunakan dalam database SQLite :

1) Membuat Database

Untuk membuat database pada SQLite, dapat dilakukan dengan cara langsung mengakses file `sqlite3.exe` dan menuliskan nama database.

Jika tidak ditemukan database tersebut maka SQLite akan otomatis membuatnya, jika ditemukan SQLite akan mengakses database tersebut dan masuk ke dalam console SQLite.

```
D:\sqlite>sqlite3 db_anggaran
SQLite version 3.6.11
Enter ".help" for instructions
Enter SQL statements terminated with a ";"
sqlite>
```

2) Menghapus Database

Menghapus database dapat dilakukan dengan cara menghapus file, SQLite bersifat flat file sehingga database tersimpan dalam satu file.

3) Membuat Tabel

Perintah untuk membuat tabel di SQLite memiliki struktur sebagai berikut :

```
create table nama_tabel (nama_field tipe_data(ukuran));
```

Berikut perintah untuk membuat tabel `t_dept` dengan field `kddept C(3) dan nmdept varchar(200)` :

```
sqlite> create table t_dept(kddept char(3), nmdept varchar(200));
```

4) Menghapus Tabel

Perintah untuk menghapus tabel di SQLite memiliki struktur sebagai berikut :

```
sqlite> drop table nama_tabel;
```

Berikut perintah untuk menghapus tabel `t_dept` :

```
Sqlite> drop table t_dept;
```

5) Menambah Field

Perintah untuk menambah field pada tabel SQLite memiliki struktur sebagai berikut :

```
sqlite> alter table nama_tabel add nama_field tipe_data(ukuran)
```

Berikut perintah untuk menambah field updater dengan tipe data character pada tabel `t_dept` :

```
sqlite> alter table t_dept add updater char(25);
```

6) Mengubah Nama Tabel

Perintah untuk mengubah nama tabel pada SQLite memiliki structure sebagai berikut :

```
sqlite> alter table nama_tabel rename to nama_tabel_baru;
```

Berikut perintah untuk mengubah nama tabel `t_dept` menjadi `ref_dept` :

```
sqlite> alter table t_dept rename to ref_dept;
```

7) Menghapus Field

Tidak ada perintah untuk menghapus field pada database SQLite

8) Input Field

Perintah untuk melakukan input data pada SQLite memiliki struktur sebagai berikut :

```
sqlite> insert into nama_tabel values (nilai_1,nilai_2);
```

Berikut contoh perintah untuk memasukkan data pada tabel ref_dept :

```
sqlite> insert into ref_dept values ('004','B P K','didik');
```

Hasil perintah tersebut dapat dilihat sebagai berikut :

```
sqlite> select * from ref_dept;
001|MAJELIS PERMUSYAWARATAN RAKYAT|
002|DEWAN PERWAKILAN RAKYAT|
004|B P K|didik
sqlite>
```

9) Menghapus Data

Perintah untuk menghapus data pada SQLite memiliki struktur sebagai berikut :

```
sqlite> delete from nama_tabel [where kondisi]
```

Kondisi yang digunakan adalah sebagaimana perintah SQL pada umumnya. Berikut contoh menghapus data dengan kode 004 pada tabel `ref_dept` :

```
sqlite> delete from ref_dept where kddept ='004';
```

10) Mengubah Data

Perintah untuk mengubah data pada SQLite memiliki struktur sebagai berikut :

```
sqlite> update nama_tabel set nama_field1 = nilai1, nama_field2 = nilai2
[where kondisi]
```

Berikut contoh mengubah field updater untuk data pada tabel `ref_dept` :

```
sqlite> update ref_dept set updater ='didik';
```

11) Menampilkan Data

Perintah untuk menampilkan data pada SQLite memiliki struktur sebagai berikut :

```
sqlite> select [nama_field|*] from nama_tabel [where kondisi]
```

Berikut contoh perintah untuk menampilkan data tabel `ref_dept` :

```
sqlite> select * from ref_dept;
001|MAJELIS PERMUSYAWARATAN RAKYAT|didik
002|DEWAN PERWAKILAN RAKYAT|didik
sqlite>
```

12) Relasi

SQLite mengenal relasi sederhana *Natural Join* dengan struktur perintah sebagai berikut :

```
sqlite> select * from nama_tabel1 join nama_tabel2 on (nama_tabel2.field =
nama_tabel1.field)
```

Berikut contoh relasi antara tabel ref_dept dengan tabel ref_unit :

```
sqlite> select * from ref_dept join ref_unit on (ref_dept.kddept =
ref_unit.kddept);
001|MAJELIS PERMUSYAWARATAN RAKYAT|didik|001|01|SEKRETARIAT
002|DEWAN PERWAKILAN RAKYAT|didik|002|01|SEKRETARIAT JENDERAL
sqlite>
```

Saat ini SQLite tidak mendukung relasi *right join* dan relasi *Full Outer Join*.

13) Penggunaan Kondisi

Penggunaan kondisi *where* dalam SQLite adalah identik dengan penggunaan kondisi dalam database MySQL. Berikut contoh penggunaan kondisi untuk pencarian data dengan menggunakan perintah `SELECT` :

```
sqlite> select * from ref_dept where nmdept like "%RAKYAT%";
001|MAJELIS PERMUSYAWARATAN RAKYAT|didik
002|DEWAN PERWAKILAN RAKYAT|didik
sqlite>
```

Hasil yang diperoleh adalah data yang mengandung kata RAKYAT pada field nama departemen (`nmdept`).

Implementasi

SQLite selain bersifat *portable* juga bersifat *server less database*, dimana SQLite tidak memerlukan server tersendiri untuk dapat menjalankan fungsinya sehingga sangat cocok digunakan sebagai database untuk aplikasi *mobile device*, pengembangan web site berskala kecil hingga menengah, hingga pengembangan aplikasi berskala *enterprise* sebagai database pada aplikasi yang bersifat *prototype* atau demo.

Salah satu contoh implementasi SQLite adalah dalam rangka pembuatan interaktif DVD Lampiran Keputusan Presiden tentang Rincian Anggaran Belanja Pemerintah Pusat (RABPP). Lampiran Keppres RABPP terdiri atas :

a) Lampiran I, berupa Rincian Anggaran Belanja Pemerintah Pusat menurut organisasi/bagian anggaran, unit organisasi, fungsi, sub fungsi, program, kegiatan, jenis belanja, dan sumber dana;

b) Lampiran II, berupa Rincian Anggaran Belanja Pemerintah Pusat menurut organisasi/bagian anggaran, unit organisasi, pusat, daerah, dan kode kewenangan;

c) Lampiran III, berupa Rincian Anggaran Belanja Pemerintah Pusat menurut organisasi/ bagian anggaran, unit organisasi, program, kegiatan, dan prakiraan maju;

d) Lampiran IV, berupa Rincian Anggaran Belanja Pemerintah Pusat menurut organisasi/ bagian anggaran, unit organisasi, dan satuan kerja.

Entitas terkecil dalam lampiran Keppres RABPP adalah satuan kerja (Satker) yang berjumlah lebih dari 23.000 Satker, berupa Kertas Kerja Satker yang berisi rincian belanja satker bersangkutan. Keseluruhan Lampiran Keppres RABPP didistribusikan kepada pihak-pihak terkait, diantaranya kepada seluruh anggota MPR, DPR, dan DPD.

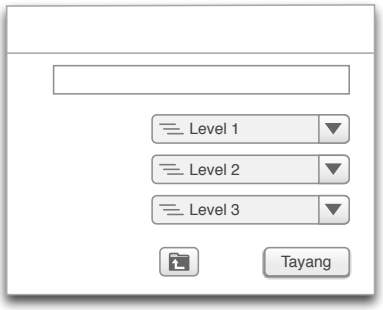
Sebelum penggunaan media DVD sebagai sarana distribusi maka *hardcopy* Lampiran RABPP-lah yang didistribusikan, sehingga diperlukan pencetakan dan penggandan dokumen sebanyak jumlah lembar Kertas Kerja Satker dikalikan dengan jumlah pihak penerima dokumen. Dengan penggunaan media DVD maka keseluruhan dokumen tersebut dapat ditampung dalam satu keping DVD ukuran kecil.

Sebuah DVD lampiran Keppres RABPP terdiri atas folder *repository* file untuk menampung seluruh *softcopy* Kertas Kerja Satker, sebuah sistem aplikasi sebagai antar muka (*interface*) antara user dan *softcopy* dokumen, dan sistem database untuk menampung seluruh data dan informasi/referensi yang diperlukan oleh sistem aplikasi. Keseluruhan sistem tersebut harus dapat berjalan pada sebuah keping DVD dengan batasan akses *Read Only*.

Sistem database diperlukan karena dalam DVD lampiran Keppres RABPP harus dimungkinkan untuk dapat melakukan proses pencarian data dan dapat menampilkan antar muka berdasarkan struktur hierarki tertentu. SQLite yang bersifat *portable* sangat memungkinkan untuk digunakan dalam kasus ini.

Persiapan database dilakukan sebagai berikut : database dan struktur tabel yang akan digunakan dibuat terlebih dahulu, selanjutnya data pada setiap tabel diisi dengan menggunakan data yang bersumber dari server database. Hasilnya adalah folder yang merupakan database dan tabel-tabel dalam folder tersebut yang berupa *flat file.* Selanjutnya database yang telah siap disertai dengan *library* SQLite, folder repository softcopy beserta isinya, dan sistem Aplikasi yang telah diuji dapat dipindahkan kedalam DVD untuk diuji sebelum dilakukan penggandan.

Contoh antar muka dengan struktur hierarki pada DVD lampiran Keppres RABPP :



Jika level 1 adalah Bagian Anggaran, dan level 2 adalah unit organisasi maka unit organisasi yang ditampilkan pada level2 adalah unit organisasi yang berada dalam kewenangan Bagian Anggaran pada level 1 dan jika level 3 adalah Satker maka data yang ditampilkan adalah Satker yang berada dalam kewenangan unit organisasi (level 2).

Sistem Aplikasi yang digunakan sebagai antar muka akan mencari data yang bersesuaian pada data dan referensi yang tersimpan pada sistem database (SQLite), selanjutnya hasil pencarian data berupa kombinasi kode akan digunakan untuk mengakses file PDF yang tersimpan pada Folder *repository softcopy* Kertas Kerja Satker untuk selanjutnya ditampilkan kepada user. Demikian mekanisme kerja interaktif DVD Lampiran Keppres RABPP yang menggunakan SQLite sebagai sistem database didalamnya.

Semoga tulisan ini bermanfaat bagi kita semua, terima kasih.

Referensi

1. Mike Chirico, "*SQLite Tutorial*", www.Freshmeat.net, 2004
2. "*Command Line Shell for SQLite*", www.SQLite.org

Biografi Penulis

Didik Setiawan
Pranata Komputer Kementerian Keuangan RI