Form dan Report sederhana dengan SQLiteBoy

- Referensi Lengkap
- Tutorial Integrasi SQLiteBoy dan OpenERP
- Tutorial Medical Record sederhana
- Dasar-dasar Python

Noprianto

Revisi 2

Tentang Buku

Form dan report sederhana dengan SQLiteBoy
(c) Noprianto <nop@tedut.com>
2013-2014
Revisi: 2 (21 April 2014, publikasi sendiri)

Tentang Aplikasi

SQLiteBoy
Aplikasi manager SQLite berbasis web dan form/report sederhana
(c) Noprianto <nop@tedut.com>
2012-2014
Lisensi: GPL

SQLiteBoy merupakan produk independen, dikembangkan terpisah dari pustaka inti SQLite, yang dimaintain oleh SQLite.org. SQLiteBoy.com ataupun SQLite.org tidak memiliki tanggung jawab apapun atas pekerjaan pihak lainnya.

Aplikasi ini diharapkan dapat berguna, namun tidak menggaransi apapun.

Buku lain oleh penulis

Kategori	Judul	Detil
Buku	Python dan Pemrograman Linux	Tahun 2002 441 halaman Penerbit Andi Yogyakarta
Buku	Panduan Praktis Debian GNU/Linux 3.1	Tahun 2006 263 halaman Penerbit Dian Rakyat / Majalah InfoLinux
Buku	OpenOffice.org 2.0	Tahun 2006 130 halaman Penerbit Dian Rakyat / Majalah InfoLinux
Buku Elektronik	wxWidgets	330+ halaman SXW/PDF
Materi Training	Python Dasar	35+ halaman ODT/PDT
Materi Training	Python GTK	50+ halaman ODT/PDF
Materi Training	Python Database	12+ halaman ODT/PDF
Materi Training	Python CGI	15+ halaman ODT/PDF

Untuk Buku Elektronik dan Materi Training, kunjungilah http://tedut.com

Informasi dan merek dagang

- OpenDocument dikembangkan oleh Organization for the Advancement of Structured Information Standards (OASIS).
- PDF, Portable Document Format merupakan format file open standard ISO 32000-1:2008, dikembangkan oleh Adobe Systems.
- SQLite adalah merek dagang dari Hipp, Wyrick & Company, Inc.
- Apache OpenOffice adalah merek dagang dari The Apache Software.
- LibreOffice adalah merek dagang dari The Document Foundation.
- Windows adalah merek dagang terdaftar dari Microsoft Corporation.
- Python adalah merek dagang dari Python Software Foundation.
- Linux adalah merek dagang terdaftar dari Linus Torvalds.
- Mac adalah merek dagang dari Apple Computer, Inc.
- web.py adalah pustaka python yang dapat digunakan untuk pengembangan aplikasi berbasis web (public domain).
- JSON adalah data interchange format, dapat dibaca di RFC 4627.
- OpenERP adalah merek dagang dari OpenERP s.a.
- ReportLab adalah pustaka python untuk bekerja dengan format PDF.
- Python Imaging Library adalah pustaka python untuk bekerja dengan berbagai format gambar.
- OpenSSL adalah merek dagang dari The OpenSSL

Software Foundation, Inc.

- pyOpenSSL adalah pustaka python untuk bekerja dengan OpenSSL.
- Mozilla FireFox adalah merek dagang dari Mozilla Foundation.

Lisensi GPL

Untuk informasi selengkapnya tentang lisensi GPL, kunjungilah alamat berikut:

http://www.gnu.org/licenses/gpl.html

Daftar Isi

1. Mengenal SQLiteBoy

Apa yang Anda lakukan apabila membutuhkan form atau data entry sederhana? Sebagai contoh, apabila mengelola keluar masuk barang gudang, pada saat masuk barang, Anda mungkin ingin mencatatkan kode barang, nomor surat jalan dari supplier, jumlah barang masuk dan berbagai informasi lainnya.

Cara yang cepat barangkali menggunakan spreadsheet seperti Apache OpenOffice Calc atau LibreOffice Calc. Anda dapat membuat tabel master barang, kemudian tabel keluar masuk barang. Untuk setiap aktifitas, Anda cukup menambahkan ke tabel yang sesuai. Walau demikian, ketika data sudah membesar, butuh diolah, atau pencatatan perlu diserahkan ke pihak lain (yang tidak perlu mengetahui semua data), kesulitan mulai datang. Data yang besar akan menjadikan operasional menjadi lebih lambat dan tampilan menjadi rumit (dan mudah terjadi kecelakaan seperti tanpa sengaja menghapus baris/kolom/sel). Pengolahan mungkin membutuhkan fungsi yang rumit atau bahkan macro. Proteksi data juga tidaklah mudah untuk diterapkan.

Anda mungkin dapat berpindah ke solusi lain seperti Apache OpenOffice Base atau LibreOffice Base, yang lebih cocok. Tabel master dan keluar masuk barang dapat dibuat. Form untuk data entry juga dapat dibangun. Anda juga dapat mengimplementasikan proteksi data tertentu. Operasional tidak lagi lambat dan tampilan tidak lagi rumit, karena tidak semua data perlu ditampilkan sekaligus. Pengolahan

data juga dapat dipermudah menggunakan berbagai fungsi yang telah disediakan. Walau demikian, ketika Anda membutuhkan akses data entry dari web (menggunakan komputer, tablet, atau ponsel) secara multi user, maka sekali lagi, kesulitan mulai datang. Apache OpenOffice Base atau LibreOffice Base perlu diinstall ke komputer, agar solusi data entry bisa digunakan.

Membangun sendiri program data entry berbasis web atau menggunakan paket program ERP juga tentu bisa menjadi solusi. Tapi, Anda bisa bayangkan kerepotan dan biaya yang diperlukan.

Maka, di sinilah SQLiteBoy hadir. Tentu saja, program ini jauh lebih sederhana dan tidak ada apa-apanya dibandingkan dengan spreadsheet dan program database yang dibahas sebelumnya, yang sejujurnya, telah menginspirasi hadirnya SQLiteBoy. Program ini juga lebih terbatas apabila dibandingkan dengan program yang dibangun sendiri ataupun paket program ERP, dimana Anda lebih bebas menyesuaikan dengan kebutuhan yang spesifik.

Tapi, barangkali Anda ingin memberikan kesempatan kepada SQLiteBoy. Program ini dapat digunakan untuk membangun solusi yang membutuhkan form/data entry dan report sederhana, dengan relatif mudah dan cepat. SQLiteBoy juga dapat digunakan sebagai management tool berbasis web untuk database SQLite.

Apabila Anda menggunakan Microsoft Windows, Anda bisa gunakan aplikasi standalone / portable / yang dapat dijalankan dari USB Flash Disk: http://tedut.com/sqliteboy.exe (satu file program

sekitar 6 MB; tidak ada instalasi apapun yang perlu dilakukan di sistem Anda; dapat dijalankan oleh user biasa). Cara pakainya juga sangat mudah: cukup klik ganda sqliteboy.exe dan dialog untuk memilih file database akan ditampilkan (Anda dapat membuat database baru). Tidak perlu mengetikkan perintah di command prompt/terminal.

Ketika dijalankan, SQLiteBoy akan membuat database baru (atau membuka database yang sudah ada), kemudian menjalankan fungsi web server (port 11738 secara default), di mana selanjutnya, Anda cukup bekerja menggunakan web browser (komputer, tablet, ponsel) dari manapun Anda terhubung. Apabila diperlukan, koneksi lewat HTTPS juga didukung.

Apabila Anda memilih untuk menjalankan dari source code, maka Anda hanya membutuhkan python 2.x (versi 2.7 atau yang lebih baru, http://www.python.org), dan satu pustaka tambahan web.py (python murni, http://webpy.org) terinstall di sistem. SQLiteBoy hanya terdiri dari satu file script python. Python sendiri tersedia untuk Windows, Linux dan Mac. Bahkan, di kedua sistem yang disebutkan terakhir, python umumnya telah tersedia secara default.

SQLiteBoy merupakan program free/open source dan dilisensikan GPL. Anda dengan bebas dapat menggunakan, mempelajari, memodifikasi ataupun menyebar luaskan program ini, sesuai aturan lisensi GPL.

Untuk informasi lebih lengkap tentang SQLiteBoy, kunjungilah http://sqliteboy.com.

SQLite dan SQLiteBoy

Sesuai namanya, SQLiteBoy bekerja dengan database SQLite (http://www.sqlite.org), sebuah database satu file (tidak membutuhkan server) yang kaya fitur dan cepat. Kunjungilah situs SQLite untuk informasi selengkapnya. Pustaka python untuk SQLite telah disertakan ke dalam python versi 2.5.

SQLiteBoy juga dapat berfungsi sebagai manager berbasis web untuk database SQLite Anda. Apabila menggunakan SQLiteBoy sebagai manager untuk database SQLite, Anda akan mendapatkan bonus berupa berbagai user-defined function (UDF) tambahan, di luar fungsi yang telah disediakan oleh SQLite. Hadirnya UDF ini, tidak akan memodifikasi database yang telah ada sebelumnya. SQLiteBoy tidak akan memodifikasi atau menambahkan apapun ke dalam database, tanpa sepengetahuan Anda.

Keamanan file database

Karena berbasis web, maka pengguna tidak memiliki akses langsung ke file database SQLite yang digunakan (akses langsung ke file sistem server tidak disediakan).

Walau demikian, tersedia fitur Query, di mana perintah SQL dapat diberikan. Oleh karenanya, data,

secara default, bisa diakses. Selain itu, SQLiteBoy juga bisa diakses lewat web browser (walaupun secara default hanya dari komputer yang menjalankan), tanpa adanya proteksi berupa user/password (SQLite tidak mengenal konsep user/password karena tidak bekerja menggunakan server).

Untuk mengatasi kedua hal tersebut, SQLiteBoy menyediakan Extended Feature, yang ketika diaktifkan oleh Anda, akan menambahkan fitur user account (dengan level standard dan admin), proteksi berupa pembatasan hak query dan akses langsung ke tabel oleh user dengan level standard dan pengaturan komputer/hosts mana saja yang boleh mengakses web SQLiteBoy.

Extended feature

Extended feature sepenuhnya opsional dan dapat diaktifkan atau dinonaktifkan oleh Anda. Apabila diaktifkan, maka SATU tabel tambahan akan dibuat:

sqliteboy

Anda dapat menghapus atau mengganti nama tabel ini, dan extended feature akan dinonaktifkan. Backuplah seperlunya, karena SEMUA data dari extended feature disimpan pada tabel ini.

Selain fitur user account dan pengaturan hosts/akses, aktifasi Extended Feature juga akan

menambahkan fitur form (data entry), report, files, notes, calculator, page, scripts dan lainnya yang akan kita bahas.

Apabila extended feature diaktifkan, maka user harus selalu login untuk bekerja dengan SQLiteBoy. Username dan password default untuk admin adalah: admin.

Pengaturan database

SQLiteBoy mendukung berbagai operasi pengaturan database berikut, tanpa perlu mengaktifkan extended feature.

- Browse: menampilkan isi tabel dengan dukungan sorting pada nama kolom (asc/desc), dengan limit pada jumlah baris yang ingin ditampilkan. Anda dapat memilih lebih dari satu baris (multiple selection), di mana baris-baris terpilih dapat dihapus atau diedit sekaligus. Baris-baris yang terpilih akan dimaintain dalam session, sehingga Anda tidak harus memilih ulang apabila Anda perlu melakukan operasi lain terlebih dahulu. Bagaimana kalau terdapat data BLOB dalam tabel? SQLiteBoy akan coba untuk mendeteksi, dan menyediakan link download (raw data). Isi table dapat pula ditampilkan dalam halaman-halaman agar tidak diload sekaligus (jumlah baris per halaman dapat diatur).
- Create: membuat tabel dengan cara mudah. Untuk setiap kolom, Anda bisa menentukan tipe, nilai

default dan apakah merupakan primary key. Nilai default bisa konstan ataupun non-konstan (seperti current_time, current_timestamp). Jumlah primary key bisa lebih dari satu.

- Query: menjalankan perintah SQL. Output dari perintah akan ditampilkan secara otomatis dalam format yang bersesuaian. Sebagai contoh, output dari perintah select akan ditampilkan dalam bentuk tabel. Sementara, output dari perintah insert akan ditampilkan sebagai bilangan apa adanya. Tentu saja, apabila terdapat kesalahan dalam perintah, pesan kesalahan juga akan ditampilkan. Hasil dari query, apabila memungkinkan, dapat pula diekspor ke file CSV. Di dalam query, kita mungkin perlu mengassign nilai ke variabel (user-defined variable): fitur ini juga disediakan (jumlah variabel per user dibatasi).
- Insert: menambahkan data ke dalam tabel. Apabila suatu kolom memiliki nilai default, maka informasi ini akan ditampilkan (Anda dapat memasukkan nilai sendiri, atau menggunakan default value tersebut). Khusus untuk kolom bertipe BLOB, maka secara otomatis, input berupa file upload akan ditampilkan.
- Edit/Update: mengedit atau mengupdate data dalam tabel. Ini berlaku untuk satu atau banyak baris. Sama seperti pada insert, edit/update dapat pula bekerja dengan nilai default dan kolom bertipe BLOB (upload dan download).
- Column: menambahkan kolom pada tabel, lengkap dengan tipe dan nilai default. Anda dapat menambahkan beberapa kolom sekaligus. SQLite tidak mendukung penghapusan/modifikasi kolom (setidaknya, pada saat program ini pertama

dikembangkan).

- Rename: mengubah nama tabel.
- Empty: mengosongkan isi tabel, tanpa menghapus tabel.
- Drop: menghapus tabel dan isinya.
- Export: ekspor tabel ke file CSV, yang nantinya dapat dibuka, diantaranya dengan spreadsheet seperti Apache OpenOffice Calc atau LibreOffice Calc.
- Import: impor tabel dari file CSV, yang sebelumnya dibuat, diantaranya dengan spreadsheet seperti Apache OpenOffice Calc atau LibreOffice Calc, atau dari file hasil ekspor.
- Schema: menampilkan perintah sql yang digunakan untuk membuat table. Dapat pula digunakan untuk membuat tabel baru berdasarkan perintah tersebut.
- Copy: mengopi isi tabel satu ke tabel lainnya, apabila terdapat kolom yang identik (nama dan tipe). Apabila diperlukan, Anda dapat mengosongkan isi tabel tujuan terlebih dahulu (empty).
- Vacuum: melakukan operasi vacuum pada database, yang akan membangun ulang keseluruhan database.

User-defined function

Untuk mempermudah bekerja dengan query, berbagai user-defined function (UDF) ditambahkan sebagai bonus, tanpa extended feature perlu diaktifkan. Untuk membedakan dengan fungsi-fungsi yang hadir

bersama SQLite, nama UDF diawali dengan:

sqliteboy_

Berbagai UDF ini dapat digunakan di Query, Form ataupun Report. Referensi dan daftar UDF dapat ditemukan di website SQLiteBoy (dan akan dibahas di buku ini), dan Anda dapat pula merequest apabila membutuhkan fungsi tertentu (walaupun tidak dapat dijanjikan bahwa request akan selalu dipenuhi).

Semua UDF yang disediakan dapat dikategorikan sebagai berikut:

- terbilang (dalam berbagai bahasa)
- pemformatan bilangan (dengan pemisah ribuan dan desimal yang bisa diatur)
- lookup tabel dengan mudah
- hash (md5, sha1 dan lainnya)
- base64 (encoding dan decoding)
- acak
- tambahan fungsi tanggal/waktu
- tambahan fungsi string
- regular expression
- berbagai utilitas

Extended feature: Form

Form merupakan data entry sederhana, yang akan tersedia apabila extended feature diaktifkan. Form dapat dibuat dengan sintaks berupa JSON (http://www.json.org). Subform sederhana juga

didukung oleh SQLiteBoy.

Form dibangun dengan konsep berupa kolom/field yang terlibat, di mana kolom/field bisa diatur agar harus diisi (required), dapat dibaca saja (readonly), telah terisi nilai tertentu (default), bebas diisi atau user harus memilih (dalam bentuk combo box) dan lainnya.

Apabila user harus memilih, maka sumber data dapat berupa nilai statik (python list) ataupun berupa hasil query.

Sementara, nilai default, dapat berupa nilai statik, hasil dari pemanggilan fungsi ataupun hasil query.

Field pada form juga mendukung event onsave, yang memungkinkan kita untuk menjalankan SQL query dan menggunakan nilainya, sebelum data form diproses. Query tersebut dapat sangat kompleks dan mengandung pemanggilan fungsi bertingkat. Sebagai contoh, Anda ingin melakukan kalkulasi tertentu atas input dari user dan menggunakan hasil kalkulasinya.

Sebelum form dapat diproses lebih lanjut, kita dapat menentukan constraint, tentang nilai yang dianggap valid, untuk setiap field. Constraint dapat pula melibatkan pemanggilan fungsi. Sebagai contoh, kita bisa menentukan bahwa suatu field harus selalu berisikan nilai bilangan lebih besar dari 100: apabila input user tidak sesuai dengan aturan tersebut, maka form tidak dapat diproses.

Anda dapat mengatur agar form digunakan untuk

menambahkan, mengupdate, menghapus data, ataupun aksi lainnya. Seperti menulis ke berbagai table atau menjalankan berbagai query yang rumit sebelum dan/atau setelah form diproses.

Tentu saja, kita bisa membatasi user mana saja yang boleh menjalankan suatu form, dengan pengaturan keamanan sederhana.

Kita bisa membuat shortcut untuk form-form yang sering diakses, agar tidak perlu memilih dari daftar form. Cukup sekali klik saja pada shortcut.

Form mendukung python handler, yang akan secara otomatis dipanggil, apabila tersedia untuk form tersebut. Python handler memudahkan integrasi dengan sistem eksternal (sebagai contoh: sistem ERP). Selain itu, python handler dapat pula berguna, sebagai contoh, dalam operasi database yang kompleks, membaca dari/menulis ke device eksternal, dan lain sebagainya. Di buku ini, kita akan membahas juga contoh tentang penggunaan python handler untuk bekerja dengan OpenERP.

Kita akan membahas referensi kode form di buku ini. Anda dapat pula membacanya di situs web SQLiteBoy.

Extended feature: Report

Report merupakan reporting/laporan sederhana, yang akan tersedia apabila extended feature diaktifkan. Report dapat dibuat dengan sintaks berupa JSON

(http://www.json.org).

Sebelum report dijalankan, user mungkin ingin memasukkan parameter tertentu, seperti tanggal awal, tanggal akhir atau kode barang. Agar hal ini dimungkinkan, report juga mendukung kolom/field sebagaimana halnya pada form, yang dibahas sebelumnya: dapat dibaca saja (readonly), telah terisi nilai tertentu (default), bebas diisi atau user harus memilih (dalam bentuk combo box) dan lainnya.

Sama seperti pada form, apabila user harus memilih, maka sumber data dapat berupa nilai statik (python list) ataupun berupa hasil query. Nilai default juga dapat berupa nilai statik, hasil dari pemanggilan fungsi ataupun hasil query.

Sebelum report dikerjakan, kita juga dapat menentukan constraint, agar report dapat dikerjakan dengan input/parameter yang dianggap valid. Sama seperti pada form, constraint juga dapat melibatkan pemanggilan fungsi.

Berbeda dengan form, report harus melibatkan sintaks SQL, dapat sangat kompleks, yang memiliki relasi dengan kolom/field yang tersedia untuk diinput oleh user.

Hasil dari query merupakan isi report. Dan, kita bebas untuk menentukan urutan kolom dalam report.

Header dan footer juga dapat dibuat, yang masing-masing bisa lebih dari satu baris, dan untuk setiap barisnya, bisa lebih dari satu kolom. Header dan

footer dapat berisikan teks statik, hasil dari query, gambar dan lainnya.

Format report yang didukung adalah: PDF, HTML, HTML (printer friendly) dan CSV.

Seperti pada form, kita bisa membatasi user mana saja yang boleh menjalankan suatu report, dengan pengaturan keamanan sederhana.

Kita bisa membuat shortcut untuk report-report yang sering diakses, agar tidak perlu memilih dari daftar report. Cukup sekali klik saja pada shortcut.

Report mendukung python handler, yang akan secara otomatis dipanggil, apabila tersedia untuk report tersebut. Python handler memudahkan integrasi dengan sistem eksternal (sebagai contoh: sistem ERP). Selain itu, python handler dapat pula berguna, sebagai contoh, dalam operasi database yang kompleks, membaca dari/menulis ke device eksternal, dan lain sebagainya. Berbeda dengan form, nilai kembalian dari python handler akan menggantikan hasil dari query. Di buku ini, kita akan membahas juga contoh tentang penggunaan python handler untuk bekerja dengan OpenERP.

Di SQLiteBoy, segala sesuatu yang dimaksudkan untuk dicetak, dianggap sebagai report. Ini termasuk report pada umumnya, nota, surat jalan, kwitansi, resep obat dan lainnya.

Kita akan membahas referensi kode report di buku ini. Anda dapat pula membacanya di situs web

SQLiteBoy.

Extended feature: User account

User account akan tersedia apabila extended feature diaktifkan. Terdapat dua level user: admin (akses penuh) dan standard (akses terbatas). Anda dapat pula membuat lebih dari satu user dengan level admin.

Setiap user yang login dapat mengganti password mereka sendiri.

Extended feature: Profile

Profile akan tersedia apabila extended feature diaktifkan. Dengan profile, setiap user bisa mengatur profile masing-masing.

Tersedia pula fitur user-defined profile, yang memungkinkan berbagai custom field pada profile user (ditentukan oleh admin atau script). Ini sangat berguna, sebagai contoh, apabila bekerja dalam lingkungan multi-company.

Extended feature: Files

Files akan tersedia apabila extended feature diaktifkan. Dengan files, user dapat mengupload berbagai file yang dimiliki. Ukuran dan jumlah maksimum file untuk user standard bisa diatur oleh admin (admin sendiri tidak dibatasi).

Setelah file diupload, user dapat mengakses dengan melihat langsung di browser (apabila didukung) atau mendownload. Setiap file milik user satu tidak dapat diakses oleh user lainnya, kecuali telah dishare.

Semua file yang telah diupload akan disimpan di dalam database. Tidak ada yang disimpan pada file sistem server.

Ketika melakukan backup, cukup satu file database saja yang dibackup. Semua telah tersimpan di dalam database tersebut.

Extended feature: Notes

Notes akan tersedia apabila extended feature diaktifkan. Dengan notes, user dapat membuat aneka catatan sederhana.

Extended feature: Calculator

Kalkulator akan tersedia apabila extended feature diaktifkan. Dengan kalkulator, user dapat melakukan kalkulasi dengan ekspresi sederhana, dengan set karakter dan panjang ekspresi tertentu.

Kita akan membahas referensi kalkulator di buku ini. Anda dapat pula membacanya di situs web SQLiteBoy.

Extended feature: Page

Page akan tersedia apabila extended feature diaktifkan. Dengan page, setiap user bisa memiliki satu halaman home page. Isi page dapat dibuat dengan teks biasa, tanpa sintaks apapun.

Pemformatan sederhana pada teks didukung.

Kita akan membahas referensi page di buku ini. Anda dapat pula membacanya di situs web SQLiteBoy.

Extended feature: Hosts

Apabila extended feature diaktifkan, maka kita bisa mengatur komputer/host mana saja yang diijinkan untuk akses ke web SQLiteBoy. Tanpa extended feature, maka hanya komputer yang menjalankan

SQLiteBoy saja (local), yang diijinkan untuk akses.

Extended feature: Script

Script akan tersedia apabila extended feature diaktifkan. Dengan script, berbagai tabel (termasuk penambahan kolom-kolom untuk tabel yang telah ada), form, report atau profile dapat dibuat secara otomatis.

Hal ini memungkinkan solusi berbasis SQLiteBoy dapat dideploy atau dishare dalam bentuk script, yang dapat diupload dan dijalankan oleh admin.

Script sendiri menggunakan kode berupa JSON (http://www.json.org) dan disimpan dalam satu file.

Kita akan membahas referensi script di buku ini. Anda dapat pula membacanya di situs web SQLiteBoy.

Extended feature: Backup

Backup akan tersedia apabila extended feature diaktifkan. Dengan backup, seorang admin dapat mendownload keseluruhan file database, lengkap dengan informasi tanggal/waktu dan nama database pada nama file backup, untuk keperluan backup.

Custom template

Tampilan/template builtin diusahakan agar dapat berjalan di berbagai browser. Template tersebut menggunakan HTML5 dan minim JavaScript, yang hanya digunakan untuk toggle select all dan dialog konfirmasi. Template ini datang dengan beberapa style (warna dan lainnya), yang dapat Anda pilih sesuai selera.

Anda, tentu saja bisa membangun template sendiri, dengan membuat file sqliteboy.html di direktori aktif (contoh template bisa dimulai dari variabel T_BASE).

Dukungan komersial

Untuk daftar penyedia dukungan komersial (termasuk mendaftarkan diri sebagai penyedia jasa), kunjungilah:
https://github.com/nopri/sqliteboy/wiki/Commercial-support

2. Latar Belakang

SQLiteBoy dikembangkan karena kebutuhan-kebutuhan berikut ini:

- Form dan report sederhana.
- Berbasis web dan dapat diakses dari perangkat mobile.
- Multi-user (level admin dan standard), dan terdapat pembatasan akses (keamanan).
- Server harus dapat dijalankan standalone (datang dengan web server sendiri), oleh user biasa di sistem.
- Berjalan di berbagai sistem operasi.
- Di Microsoft Windows, program harus terdiri dari satu file executable, yang dapat dijalankan dari removable storage, tanpa instalasi system-wide.
- Database harus terdiri dari satu file dan semua (gambar, file upload dan lainnya) harus disimpan dalam database tersebut (tidak di file system).
- Definisi form, report dan lainnya harus sederhana dan kompatibel (antar versi). Apabila terdapat perubahan secara internal, kode yang telah dibuat pada versi sebelumnya harus selalu dapat digunakan pada versi-versi setelahnya.
- Harus dapat diintegrasikan dengan sistem eksternal (contoh: ERP).
- Fungsi-fungsi bantu (terbilang dalam beberapa

bahasa, format bilangan, tanggal/waktu, dan lainnya) harus sudah tersedia. Fungsi-fungsi ini harus diimplementasikan pada level database sehingga dapat pula digunakan pada SQL Query.

- Program harus sekaligus berfungsi sebagai management tool untuk database yang digunakan.
- Solusi dapat di-deploy dalam bentuk script.
- Tampilan dapat dikustomisasi oleh user tanpa mengubah kode program.

3. Instalasi / Cara menjalankan

SQLiteBoy dirancang untuk sangat mudah diinstall dan dijalankan. Selain itu, tidak ada file konfigurasi yang perlu dibuat dan dapat dijalankan oleh user biasa (pada port yang diijinkan).

Apabila Anda menggunakan sistem operasi Microsoft Windows, SQLiteBoy juga tersedia dalam satu file executable standalone (http://tedut.com/sqliteboy.exe), yang langsung dapat dijalankan. Tidak ada software lain apapun yang perlu diinstall. Tidak ada command line yang perlu diketik (dialog akan digunakan). Anda bisa jalankan dari USB Flash Disk dan membawanya program / database-nya bersama Anda. Semua fungsionalitas terkandung di dalam executable berukuran sekitar 6 MB, dilengkapi source code, dokumentasi lengkap dan dukungan SSL.

Apabila Anda memilih untuk menjalankan dari source code, maka pada dasarnya, hanya python 2.x (http://www.python.org) dan web.py (http://webpy.org) yang harus diinstall. Python sendiri tersedia untuk berbagai sistem operasi dan umum ditemukan terinstall secara default. Sementara, web.py sendiri adalah pustaka yang dikembangkan dengan python murni, dan dapat diinstall antara lain dengan cara mengopikan sebuah direktori. SQLiteBoy menggunakan pendekatan membutuhkan sesedikit mungkin pustaka tambahan (diluar bawaan python). Apabila ada fungsi yang membutuhkan pustaka tambahan tertentu, maka harus

dibuat opsional: program tetap dapat berjalan dan fungsi tersebut baru diaktifkan apabila pustaka tersebut ditemukan di sistem.

SQLiteBoy dikembangkan dan diuji pada python versi 2.7. Apabila ditemukan istilah python 2.x di dalam buku ini, maka dimaksudkan cpython 2 versi 2.7 atau yang lebih baru (python 2.6 mungkin atau mungkin tidak dapat digunakan).

Source code

Source code SQLiteBoy hanya terdiri dari satu file: sqliteboy.py. Dokumentasi lengkap disimpan dalam file README.rst.

Untuk mendownload source code, kunjungilah situs web SQLiteBoy.

Atau, apabila menggunakan git, Anda dapat melakukan clone dengan perintah berikut:

git clone https://github.com/nopri/sqliteboy.git

Kebutuhan sistem

Apabila Anda menggunakan source code, berikut adalah apa yang dibutuhkan untuk menjalankan SQLiteBoy dengan fungsionalitas penuh.

- Python 2.x, disarankan versi 2.7 atau yang lebih baru.
- web.py, disarankan versi 0.37. Apabila Anda menggunakan sistem operasi Linux, cukup banyak distribusi yang telah memaketkan web.py.
- Modul sqlite3, disertakan sejak python versi 2.5. Apabila Anda menggunakan sistem operasi Linux, pastikanlah dukungan sqlite3 telah disertakan di dalam paket python oleh distribusi yang Anda gunakan (apabila tidak, Anda mungkin perlu menginstall paket tambahan).
- Modul json, disertakan sejak python versi 2.6. Apabila Anda menggunakan sistem operasi Linux, pastikanlah dukungan json telah disertakan di dalam paket python oleh distribusi yang Anda gunakan (apabila tidak, Anda mungkin perlu menginstall paket tambahan).
- Opsional: ReportLab (http://www.reportlab.com) dan Python Imaging Library (http://www.pythonware.com/products/pil/), untuk output report dalam format PDF. Kedua pustaka ini sangat umum tersedia dalam berbagai distribusi Linux. Untuk sistem operasi Microsoft Windows, juga tersedia installer untuk kedua pustaka tersebut.
- Opsional: pyOpenSSL (https://pypi.python.org/pypi/pyOpenSSL), untuk dukungan SSL. Pustaka ini sangat umum tersedia dalam berbagai distribusi Linux, dan juga tersedia installer untuk sistem operasi Microsoft Windows.

Cara menjalankan

Apabila menggunakan sqliteboy.exe, maka cara termudah untuk menjalankan adalah dengan klik ganda pada file tersebut. Dialog pemilihan file akan ditampilkan, di mana kita bisa memilih atau memasukkan nama file database (termasuk file yang belum ada).

Akan tetapi, ada kalanya, kita ingin mengatur port yang akan digunakan atau meredireksi standard output/standard error ke file log tertentu. Oleh karena itu, kita dapat memberikan argumen ketika menjalankan SQLiteBoy, lewat command line interface.

Berikanlah perintah sesuai pola berikut:

sqliteboy.exe <database_file> [port]

atau

python sqliteboy.py <database_file> [port]
(apabila menggunakan source code)

Sebagai penjelasan:

- <database_file> adalah path ke file database. SQLiteBoy akan mencoba untuk melakukan koneksi SQLite ke file tersebut. Apabila gagal (misal karena file tidak dapat dibaca atau bukan merupakan database SQLite), maka pesan kesalahan akan ditampilkan. Apabila file tidak ditemukan sebelumnya, maka file baru akan dicoba untuk dibuat.

- [port] adalah nomor port di sistem, di mana HTTP Server yang datang bersama SQLiteBoy akan dijalankan. Secara default, port adalah 11738.

Contoh (semua perintah diketikkan dalam satu baris):

```
sqliteboy.exe a.db
sqliteboy.exe a.db 8000
```

atau

```
python sqliteboy.py a.db
python sqliteboy.py a.db 8000
```

atau

```
python sqliteboy.py a.db 8000 >out.txt 2>err.txt &
```

(menggunakan source code, pada sh-compatible shell dengan job-control, dan ingin menjalankan di background. Standard output akan diredireksi ke file out.txt dan standard error akan diredireksi ke file err.txt. Apabila tersedia di sistem operasi yang digunakan, Anda bisa pula menggunakan /dev/null apabila tidak memerlukan log).

Setelah SQLiteBoy berjalan, gunakanlah web browser (contoh: Mozilla Firefox) untuk mengunjungi http://localhost:11738, atau http://localhost:PORT, apabila menggunakan port lainnya. Gantilah http dengan https apabila menggunakan SSL (kita akan membahasnya lebih lanjut di bagian lain, di dalam buku ini).

Extended feature

Ketika SQLiteBoy dijalankan dengan database yang belum diaktifkan extended feature-nya, maka hanya fungsi management tool SQLite dan user-defined function yang tersedia. Selain itu, web SQLiteBoy hanya dapat diakses dari komputer lokal (yang menjalankan SQLiteBoy).

Untuk memaksimalkan fungsi SQLiteBoy, ada baiknya juga untuk mengaktifkan extended feature, dengan cara klik pada link:

create _sqliteboy_ table and enable extended features

yang dapat ditemukan di halaman depan (/). User dan password default adalah admin.

Direktori aktif

Kita tidak harus bekerja dari/di dalam direktori instalasi SQLiteBoy. Kita bisa bekerja dari direktori lain, di mana direktori lain tersebut akan menjadi direktori aktif (current working directory).

Di dalam direktori aktif tersebut, kita bisa tempatkan SSL Certificate/Private Key, python

handler, custom template atau lainnya, tanpa mengotori direktori instalasi SQLiteBoy. Dengan demikian pula, kita bisa memiliki satu kopi instalasi SQLiteBoy, namun dengan banyak database dan python handlernya.

Sebagai contoh, andaikata SQLiteBoy dikopikan di /opt/sqliteboy, Anda bisa bekerja dari /home/user:

```
cd /home/user
python /opt/sqliteboy/sqliteboy.py a.db
```

Dalam contoh tersebut, maka /home/user adalah direktori aktif.

User lain, sebagai contoh, dapat pula bekerja dari /home/user2 dan segala file seperti SSL Certificate/Private Key, python handler, custom template dan lainnya terpisah dari /home/user.

Hal ini, selain menghemat ruang kosong di file system, juga akan memudahkan proses update, karena cukup satu instalasi saja yang perlu diupdate.

Materi tentang SSL, python handler dan custom template akan dibahas di bagian lain, di dalam buku ini.

Catatan: nama tabel

SQLiteBoy tidak dapat bekerja dengan nama tabel (bukan file database) yang mengandung spasi.

4. Dukungan SSL

Apabila SQLiteBoy dijalankan pada server tersendiri dan pengguna mengakses SQLiteBoy lewat jaringan (contoh: internet), maka ada baiknya kita menggunakan SSL (Secure Sockets Layer), agar data dienkripsi terlebih dahulu sebelum dikirim. Hal ini akan mempersulit pihak yang ingin mencuri data ketika data sedang melewati jaringan.

Untuk mengaktifkan dukungan SSL pada SQLiteBoy, kita memerlukan SSL Certificate, yang dapat dibeli dari berbagai Certificate Authority (atau resellernya), ataupun dibuat sendiri (self-signed).

Apabila SQLiteBoy dijalankan dari jaringan internal kantor, maka seharusnya SSL Certificate yang dibuat sendiri sudah memadai. Memang, pada saat mengunjungi lewat web browser, akan terdapat peringatan bahwa SSL Certificate yang digunakan tidak dapat divalidasi, namun fungsi keamanan data tetap kita dapatkan.

Apabila SQLiteBoy dijalankan dari jaringan internet, pada server dengan IP Publik Statik, maka ada baiknya kita membeli SSL Certificate, yang dapat divalidasi sehingga tidak ada pesan peringatan yang mungkin membuat pengguna bingung atau khawatir.

Setelah SSL Certificate dibeli atau dibuat, tempatkanlah file SSL Certificate dan SSL Private Key di dalam direktori aktif. Nama file yang

digunakan harus menuruti aturan berikut:

- sqliteboy.cert untuk SSL Certificate
- sqliteboy.key untuk SSL Private Key

Ketika SQLiteBoy dijalankan, dan SSL Certificate bisa diload (karena pustaka tersedia dan Certificate/Private Key valid), maka ketika mengunjungi SqliteBoy, kita menggunakan protokol https.

Apabila dukungan SSL diaktifkan dan kita mengunjungi lewat protokol http, maka pesan kesalahan akan ditampilkan (server akan tetap merespon).

Self-signed Certificate

Untuk mencoba, kita dapat membuat self-signed certificate sendiri. Apa yang kita perlukan adalah program openssl (http://www.openssl.org), yang sangat umum terinstall default pada berbagai distribusi Linux. Di sistem operasi Microsoft Windows, kita dapat menggunakan openssl yang datang bersama distribusi git.

Perintah berikut bisa digunakan (diketikkan dalam satu baris):

```
openssl req -new -x509 -newkey rsa:1024 -keyout sqliteboy.key -out sqliteboy.cert -days 365 -nodes
```

Beberapa pertanyaan akan diajukan, dan setelah kita

menjawab, maka SSL Certificate dan SSL Private Key akan dibuat.

5. Custom Template

Walaupun style dapat diatur, tampilan user interface/template default SQLiteBoy boleh dikatakan cukup sederhana.

Apabila Anda butuh untuk melakukan kustomisasi, buatlah sebuah file sqliteboy.html dan tempatkan di dalam direktori aktif ketika menjalankan SQLiteBoy.

Untuk isi file, cara termudah adalah dengan mengopi isi variabel T_BASE dalam source code sqliteboy.py.

Pastikan isi template tersebut tidak diawali dengan baris:

```
$def with (data, content)
```

sebagaimana yang diharapkan untuk template pada web.py. Baris tersebut akan ditambahkan secara otomatis, ketika membaca isi file custom template.

6. Referensi: UDF

Di bagian ini, kita akan membahas berbagai UDF (user-defined function) yang datang bersama SQLiteBoy.

Berbagai fungsi tersebut ditujukan untuk melengkapi dan saling bekerja sama dengan fungsi-fungsi yang datang bersama SQLite, dan bisa digunakan tanpa atau dengan extended feature diaktifkan.

Setiap nama fungsi akan selalu diawali dengan sqliteboy_.

Pemanggilan fungsi tunggal dilakukan dengan statemen select. Pemanggilan fungsi dapat dilakukan bertingkat.

Contoh 1:

```
select sqliteboy_number_to_words('-
12345678912345678912345678 9.123', 'id') as a
```

Hasil query adalah:

min seratus dua puluh tiga triliun empat ratus lima puluh enam milyar tujuh ratus delapan puluh sembilan juta seratus dua puluh tiga ribu empat ratus lima puluh enam triliun tujuh ratus delapan puluh sembilan milyar seratus dua puluh tiga juta empat ratus lima puluh enam ribu tujuh ratus delapan puluh sembilan koma satu dua tiga

Contoh 2:

```
select length(sqliteboy_number_to_words('-
123456789123456789123456789.123', 'id')) as a
```

Hasil query adalah:

330

Selanjutnya, untuk semua contoh penggunaan fungsi yang kita bahas, statemen select tidak dituliskan.

Catatan penting:

- Sebagian besar fungsi akan melakukan konversi parameter yang dilewatkan, dari atau ke string, sehingga secara umum, walaupun parameter mengharapkan bilangan, kita dapat melewatkan sebagai string bilangan.
- Beberapa pemanggilan fungsi yang perlu melewatkan bilangan besar bahkan harus melewatkan sebagai string. Periksalah ulang apabila fungsi mengembalikan nilai yang tidak seharusnya, padahal input telah benar.
- Fungsi yang datang diusahakan untuk tidak terlalu cerewet dan secara umum, apabila terjadi kesalahan, akan mengembalikan string kosong atau bilangan dengan makna serupa.
- Selalu merujuklah kepada referensi ini apabila diperlukan.

sqliteboy_strs(s)
Konversi ke str

sqliteboy_as_integer(s)
Konversi ke integer

sqliteboy_as_float(s)
Konversi ke float

sqliteboy_len(s)
Mengukur panjang str. Kita dapat pula menggunakan fungsi length() yang datang bersama SQLite. Dipertahankan untuk kompatibilitas.

sqliteboy_md5(s)
Menghitung MD5 hash

sqliteboy_sha1(s)
Menghitung SHA1 hash

sqliteboy_sha224(s)
Menghitung SHA224 hash

sqliteboy_sha256(s)
Menghitung SHA256 hash

sqliteboy_sha384(s)
Menghitung SHA384 hash

sqliteboy_sha512(s)
Menghitung SHA512 hash

sqliteboy_b64encode(s)
Encoding base64

sqliteboy_b64decode(s)
Decoding base64

sqliteboy_randrange(a, b)
Mendapatkan angka acak antara a dan b

sqliteboy_randstr(s, a, b)
Mendapatkan string acak dari sejumlah set karakter, dengan panjang minimum dan maksimum yang dapat ditentukan.

argumen:
- s: set karakter
- a: panjang minimum, > 0
- b: panjang maksimum, > 0, >=a

contoh:
```
sqliteboy_randstr('abcdef123456', 3, 8)
-> 'e2e6'
```

tips:
- untuk mendapatkan hasil acak dengan panjang tetap, gunakanlah a = b
- gunakanlah fungsi sqliteboy_randstr2() atau sqliteboy_randstr3() untuk set karakter yang

telah ditentukan.
- Gunakanlah fungsi sqliteboy_randstr_simple() untuk pengacakan sederhana.

sqliteboy_randstr2(a, b)
Mendapatkan string acak dari sejumlah set karakter yang telah ditentukan (huruf + digit + tanda baca), dengan panjang minimum dan maksimum yang dapat ditentukan.

argumen:
- a: panjang minimum, > 0
- b: panjang maksimum, > 0, >=a

contoh:
```
sqliteboy_randstr2(3, 8)
-> '"Z\@Z'
```

sqliteboy_randstr3(a, b)
Mendapatkan string acak dari sejumlah set karakter yang telah ditentukan (huruf + digit), dengan panjang minimum dan maksimum yang dapat ditentukan.

argumen:
- a: panjang minimum, > 0
- b: panjang maksimum, > 0, >=a

contoh:
```
sqliteboy_randstr3(3, 8)
-> 'nItJ8'
```

sqliteboy_randstr_simple()
Mendapatkan string acak (sederhana).

contoh:
```
sqliteboy_randstr_simple()
-> 'VUmDAQeJCpww9IjmiexrWRuRT6ZgpacKVdOA'
```

sqliteboy_is_datetime_format(s, fmt)
Memeriksa apakah s merupakan datetime yang valid sesuai dengan format (fmt) yang diberikan.

argumen:
- s: string input
- fmt: string format datetime

contoh:
```
sqliteboy_is_datetime_format('2014', '%Y')
-> 1

sqliteboy_is_datetime_format('2014-01-01', '%Y-%m-
%d')
-> 1

sqliteboy_is_datetime_format('2014-01-01', '%Y-%m-
%d %H:%M:%S')
-> 0

sqliteboy_is_datetime_format('2014-01-01 01:02:03',
'%Y-%m-%d %H:%M:%S')
-> 1
```

tips:
- Gunakanlah fungsi sqliteboy_is_datetime(), sqliteboy_is_date() atau sqliteboy_is_time()

untuk string format datetime yang telah ditentukan.

sqliteboy_is_datetime(s)
Memeriksa apakah s merupakan datetime yang valid (YYYY-MM-DD HH:MM:SS).

sqliteboy_is_date(s)
Memeriksa apakah s merupakan date yang valid (YYYY-MM-DD).

sqliteboy_is_time(s)
Memeriksa apakah s merupakan time yang valid (HH:MM:SS).

sqliteboy_time()
Mendapatkan jumlah detik sejak Epoch.

sqliteboy_time2(s)
Mendapatkan jumlah detik sejak Epoch, untuk datetime s (YYYY-MM-DD HH:MM:SS).

argumen:
- s: string datetime

contoh:
sqliteboy_time2('2012-03-28 19:20:21')
-> 1332937221.0

sqliteboy_time3(f)
Mendapatkan string datetime (YYYY-MM-DD HH:MM:SS) dari jumlah detik sejak Epoch, dalam local time.

argumen:
- f: jumlah detik sejak Epoch

contoh:
sqliteboy_time3(1)
-> 1970-01-01 07:00:01
-> timezone adalah UTC+7

sqliteboy_time3a()
alias dari sqliteboy_time3(sqliteboy_time())

sqliteboy_time4(f)
Mendapatkan string datetime (YYYY-MM-DD HH:MM:SS) dari jumlah detik sejak Epoch, dalam UTC.

argumen:
- f: jumlah detik sejak Epoch

contoh:
sqliteboy_time4(1)
-> 1970-01-01 00:00:01

sqliteboy_time4a()
alias dari sqliteboy_time4(sqliteboy_time())

sqliteboy_time5(s1, s2, mode)
Menghitung selisih antara dua datetime dalam detik,

menit, jam, hari atau tahun.

Satu tahun diperhitungkan sebagai 365.2425 hari.

argumen:

- s1: YYYY-MM-DD HH:MM:SS
- s2: YYYY-MM-DD HH:MM:SS
- mode: 1=detik, 2=menit, 3=jam, 4=hari, 5=tahun

contoh:

```
sqliteboy_time5('2010-11-12 13:14:15', '2011-12-13
14:15:16', 1)
-> 34218061.0

sqliteboy_time5('2010-11-12 13:14:15', '2011-12-13
14:15:16', 2)
-> 570301.016667

sqliteboy_time5('2010-11-12 13:14:15', '2011-12-13
14:15:16', 3)
-> 9505.01694444

sqliteboy_time5('2010-11-12 13:14:15', '2011-12-13
14:15:16', 4)
-> 396.042372685

sqliteboy_time5('2010-11-12 13:14:15', '2011-12-13
14:15:16', 5)
-> 1.08432718724
```

tips:

- apabila s1 atau s2 kosong atau tidak valid, maka datetime saat ini (local time) akan dipergunakan.
- Gunakanlah fungsi sqliteboy_number_format()

untuk memformat hasil kembalian fungsi.

sqliteboy_time6(f, year, month, day, mode)
Memformat selisih antara dua datetime dalam format y (tahun) m (bulan) d (hari).

argumen:
- f: selisih, dalam tahun, dengan menggunakan fungsi sqliteboy_time5() dengan mode=5.
- year: string tahun
- month: string bulan
- day: string hari
- mode: 1=30.44 hari per bulan, 2=30 hari per bulan, 3=31 hari per bulan

contoh:
```
sqliteboy_time6(sqliteboy_time5('2010-11-12
01:02:03', '2011-12-13 11:12:13', 5), ' years ', '
months ', ' days ', 0)
-> 1 years 1 months 1 days

sqliteboy_time6(sqliteboy_time5('2010-11-12
01:02:03', '2011-10-11 11:12:13', 5), ' years ', '
months ', ' days ', 0)
-> 0 years 10 months 29 days

sqliteboy_time6(sqliteboy_time5('2013-01-01
10:20:30', '2013-01-02 10:20:30', 5), ' years ', '
months ', ' days ', 0)
-> 0 years 0 months 1 days

sqliteboy_time6(sqliteboy_time5('2013-01-02
10:20:30', '2013-01-01 10:20:30', 5), ' years ', '
months ', ' days ', 0)
```

```
-> 0 years 0 months -1 days

sqliteboy_time6(1000, ' years ', ' months ', ' days
', 0)
-> 1000 years 0 months 0 days

sqliteboy_time6(1.5, ' years ', ' months ', ' days
', 0)
-> 1 years 6 months 0 days

sqliteboy_time6(1.24, ' years ', ' months ', ' days
', 0)
-> 1 years 2 months 27 days

sqliteboy_time6(1.24, ' years ', ' months ', ' days
', 1)
-> 1 years 2 months 26 days

sqliteboy_time6(1.24, ' years, ', ' months, ', '
days', 0)
-> 1 years, 2 months, 27 days

sqliteboy_time6(1.24, ' tahun ', ' bulan ', ' hari
', 0)
-> 1 tahun 2 bulan 27 hari
```

sqliteboy_is_leap(n)
Apakah n merupakan tahun kabisat.

argumen:

- n: tahun

nilai kembalian:

- 1 (tahun kabisat) atau 0 (bukan tahun kabisat)

sqliteboy_lower(s)
Mendapatkan string dalam versi huruf kecil. Kita dapat pula menggunakan fungsi lower() yang datang bersama SQLite. Dipertahankan untuk kompatibilitas.

sqliteboy_upper(s)
Mendapatkan string dalam versi huruf besar. Kita dapat pula menggunakan fungsi upper() yang datang bersama SQLite. Dipertahankan untuk kompatibilitas.

sqliteboy_swapcase(s)
Mendapatkan string dalam format huruf besar untuk setiap karakter input huruf kecil, dan sebaliknya.

sqliteboy_capitalize(s, what)
Kapitalisasi string.

argumen:
- s: string input
- what: 0=kata pertama saja, 1=semua kata

contoh:
```
sqliteboy_capitalize('hello world', 0)
-> 'Hello world'

sqliteboy_capitalize('hello world', 1)
-> 'Hello World'
```

sqliteboy_justify(s, justify, length, padding)

Memformat string rata kiri, kanan atau tengah.

argumen:
- s: string input
- justify: 0=kiri, 1=kanan, 2=tengah
- length: panjang string baru
- padding: satu karakter padding

contoh:
```
sqliteboy_justify('hello', 0, 10, 'x')
-> 'helloxxxxx'

sqliteboy_justify('hello', 1, 10, 'x')
-> 'xxxxxhello'

sqliteboy_justify('hello', 2, 10, 'x')
-> 'xxhelloxxx'

sqliteboy_justify(12345, 1, 10, 0)
-> '0000012345'
```

sqliteboy_find(s, sub, position, case)
Menemukan indeks dalam s dimana substring sub ditemukan.

argumen:
- s: string input
- sub: substring
- position: 0=indeks terendah, 1=indeks tertinggi
- case: 0=abaikan perbedaan huruf besar/kecil, 1=bedakan huruf besar/kecil

nilai kembalian:

- -1 (tidak ditemukan), > -1 (ditemukan, indeks dimulai dari 0)

contoh:
```
sqliteboy_find('hello sqliteboy', 'e', 0, 0)
-> 1

sqliteboy_find('hello sqliteboy', 'e', 1, 0)
-> 11

sqliteboy_find('hello sqlitEboy', 'e', 1, 0)
-> 11

sqliteboy_find('hello sqlitEboy', 'e', 1, 1)
-> 1
```

sqliteboy_reverse(s)
Membalik string

argumen:
- s: string input

contoh:
```
sqliteboy_reverse('hello world')
-> 'dlrow olleh'

sqliteboy_reverse(12345)
-> '54321'
```

sqliteboy_repeat(s, n)
Mengulang string s sejumlah n kali.

argumen:

- s: string input
- n: jumlah pengulangan

contoh:
```
sqliteboy_repeat('sqliteboy ', 5)
-> 'sqliteboy sqliteboy sqliteboy sqliteboy
sqliteboy'

sqliteboy_repeat(1, 20)
-> '11111111111111111111'

sqliteboy_repeat('=', 10)
-> '=========='
```

sqliteboy_count(s, sub, case)
Menghitung jumlah ditemukan substring sub dalam string s.

argumen:
- s: string input
- sub: substring
- case: 0=abaikan perbedaan huruf besar/kecil, 1=bedakan huruf besar/kecil

nilai kembalian:
- 0 (tidak ditemukan), > 0 (jumlah ditemukan)

contoh:
```
sqliteboy_count('hello sqliteboy', 'e', 0)
-> 2

sqliteboy_count('hello hello hello', 'Hello', 0)
-> 3
```

```
sqliteboy_count('hello hello hello', 'Hello', 1)
-> 0
```

sqliteboy_is_valid_email(s)
Apakah s merupakan format alamat email yang valid.

nilai kembalian:
- 1 (valid), 0 (tidak valid)

sqliteboy_match(s1, s2)
Pencocokan (match) sesuai pola regular expression.

argumen:
- s1: string pola
- s2: string yang akan diuji

nilai kembalian:
- 1 (cocok), 0 (tidak cocok)

sqliteboy_is_number(n)
Apakah n merupakan bilangan.

argumen:
- n: string atau bilangan yang akan diuji

nilai kembalian:
- 1 (bilangan), 0 (bukan bilangan)

sqliteboy_is_float(n)

Apakah n merupakan bilangan floating point.

argumen:
- n: string atau bilangan yang akan diuji

nilai kembalian:
- 1 (bilangan floating point), 0 (bukan bilangan floating point)

sqliteboy_is_integer(n)
Apakah n merupakan bilangan bulat.

argumen:
- n: string atau bilangan yang akan diuji

nilai kembalian:
- 1 (bilangan bulat), 0 (bukan bilangan bulat)

sqliteboy_normalize_separator(s, separator, remove_space, unique)
Pada string yang dipisahkan oleh separator, fungsi ini mengembalikan string dimana separator yang tidak diperlukan akan dibuang dan dimana spasi kosong yang tidak diperlukan dapat dihapus, serta tersedia opsi hanya mengambil setiap anggota yang unik saja.

argumen:
- s: string input
- separator: string separator

- remove_space: hapus spasi kosong (1 atau 0)
- unique: hanya mengambil setiap anggota yang unik saja (1 atau 0)

contoh:
```
sqliteboy_normalize_separator(',,,,,1,1,,  2,  3,
4,,,,', ',', 1, 1)
-> '1,2,3,4'
```

sqliteboy_split0(s, separator, index)
Memecah string berdasarkan separator dan kemudian mengembalikan anggota sesuai indeks (dalam list).

argumen:
- s: string input
- separator: string separator
- index: indeks anggota dalam list hasil pemecahan

nilai kembalian:
- string sesuai indeks dalam daftar, atau ''

contoh:
```
sqliteboy_split0('s.q.l.i.t.e.b.o.y', '.', 1)
-> 'q'

sqliteboy_split0('s.q.l.i.t.e.b.o.y', '', 1)
-> ''

sqliteboy_split0('s.q.l.i.t.e.b.o.y', '.', -3)
-> 'b'

sqliteboy_split0('h e l l o', '', 1)
-> 'e'
```

tips:
- apabila empty string digunakan sebagai separator, maka whitespace akan digunakan sebagai separator.

sqliteboy_chunk(s, n, separator, justify, padding)
Mengelompokkan/memecah string ke dalam bagian-bagian, yang masing-masing memiliki panjang yang sama.

argumen:
- s: string input
- n: panjang setiap bagian yang diinginkan
- separator: string separator
- justify: 0=rata kiri, 1=rata kanan
- padding: satu karakter padding

contoh:
```
select sqliteboy_chunk('123456789', 3, '-', 1, 'x')
-> '123-456-789'

select sqliteboy_chunk('123456789', 2, '-', 0, 'x')
-> '12-34-56-78-9x'

select sqliteboy_chunk('123456789', 2, '-', 1, 'x')
-> 'x1-23-45-67-89'

select sqliteboy_chunk('123456789', 4, ',', 1, '*')
-> '***1,2345,6789'
```

sqliteboy_number_format(n, decimals, decimal_point,

thousands_separator)
Memformat bilangan (atau string bilangan) dengan pengelompokan ribuan dan desimal, dimana pemisah ribuan dan pemisah desimal dapat diatur.

Dapat pula bekerja dengan bilangan dalam notasi scientific (e).

argumen:

- n: bilangan atau string bilangan. Gunakanlah string untuk bilangan yang sangat besar.
- decimals: jumlah desimal di belakang koma
- decimal_point: pemisah desimal
- thousands_separator: pemisah ribuan

contoh:

```
sqliteboy_number_format(12345, 3, '.', ',')
-> '12,345'

sqliteboy_number_format(12345, 3, ',', '.')
-> '12.345'

sqliteboy_number_format(12345.1234, 3, ',', '.')
-> '12.345,123'

sqliteboy_number_format(12345.1234, 0, ',', '.')
-> '12.345'

sqliteboy_number_format(12345.1234, 10, ',', '.')
-> '12.345,1234000000'

sqliteboy_number_format(12345.1234, 2, ',', ' ')
-> '12 345,12'

sqliteboy_number_format('-
```

```
12345678912345678912345678912345678912.123', 10,
',', '.')
-> '-
12.345.678.912.345.678.912.345.678.912.345.678.912,
1230000000'
```

sqliteboy_number_to_words(s, language)
Fungsi terbilang dalam beberapa bahasa.

Kita akan membahas referensi terbilang tentang bahasa yang didukung dan lainnya, di bagian lain dalam buku ini.

argumen:

- s: string input
- language: kode bahasa

nilai kembalian:

- kata-kata terbilang untuk bilangan, atau '' apabila bahasa tidak didukung atau terjadi kesalahan

```
contoh (kode bahasa 'id'):
sqliteboy_number_to_words('-0', 'id')
-> 'nol'

sqliteboy_number_to_words('11', 'id')
-> 'sebelas'

sqliteboy_number_to_words('1000', 'id')
-> 'seribu'

sqliteboy_number_to_words('1000000', 'id')
-> 'satu juta'
```

```
sqliteboy_number_to_words('-
12345678912345678912345678 9.123456789', 'id')
-> 'min seratus dua puluh tiga triliun empat ratus
lima puluh enam milyar tujuh ratus delapan puluh
sembilan juta seratus dua puluh tiga ribu empat
ratus lima puluh enam triliun tujuh ratus delapan
puluh sembilan milyar seratus dua puluh tiga juta
empat ratus lima puluh enam ribu tujuh ratus
delapan puluh sembilan koma satu dua tiga empat
lima enam tujuh delapan sembilan'

contoh (kode bahasa 'en1'):
sqliteboy_number_to_words('-0', 'en1')
-> 'zero'

sqliteboy_number_to_words('11', 'en1')
-> 'eleven'

sqliteboy_number_to_words('1000', 'en1')
-> 'one thousand'

sqliteboy_number_to_words('1000000', 'en1')
-> 'one million'

sqliteboy_number_to_words('-
12345678912345678912345678 9.123456789', 'en1')
-> 'minus one hundred twenty-three trillion four
hundred fifty-six billion seven hundred eighty-nine
million one hundred twenty-three thousand four
hundred fifty-six trillion seven hundred eighty-
nine billion one hundred twenty-three million four
hundred fifty-six thousand seven hundred eighty-
nine point one two three four five six seven eight
nine'
```

sqliteboy_lookup1(table, field, field1, value1, function, distinct)
Mencari ke dalam tabel dan melakukan query sesuai pola berikut:

SELECT <function>(<field>) FROM <table> WHERE <field1>=<value1>

kemudian mengembalikan nilai hasil pemanggilan fungsi.

argumen:
- table: nama tabel dalam database
- field: nama field dalam tabel
- field1: field where
- value1: value untuk field where (field1)
- function: salah satu dari avg, count, group_concat, max, min, sum, total
- distinct: 0=non distinct, 1=distinct

nilai kembalian:
- nilai hasil pemanggilan fungsi dalam string, atau '' apabila terjadi kesalahan

contoh isi tabel 'lookup':

a	**b**
a	0
a	1
a1	2
a2	3

```
contoh:
sqliteboy_lookup1('lookup', 'b', 'a', 'a', 'avg',
0)
-> '0.5'

sqliteboy_lookup1('lookup', 'a', 'a', 'a', 'count',
0)
-> '2'

sqliteboy_lookup1('lookup', 'a', 'a', 'a', 'count',
1)
-> '1'

sqliteboy_lookup1('lookup', 'a', 'a', 'a',
'group_concat', 0)
-> 'a,a'

sqliteboy_lookup1('lookup', 'b', 'a', 'a', 'max',
0)
-> '1'

sqliteboy_lookup1('lookup', 'b', 'a', 'a', 'min',
0)
-> '0'

sqliteboy_lookup1('lookup', 'b', 'a', 'a', 'sum',
0)
-> '1'

sqliteboy_lookup1('lookup', 'b', 'a', 'a2',
'total', 0)
-> '3.0'
```

sqliteboy_lookup2(table, field, field1, value1, order, default)
Mencari ke dalam tabel dan melakukan query sesuai pola berikut:

SELECT <field> FROM <table> WHERE <field1>=<value1> ORDER BY rowid asc

atau

SELECT <field> FROM <table> WHERE <field1>=<value1> ORDER BY rowid desc

kemudian mengembalikan baris pertama (atau nilai kembalian default).

argumen:
- table: nama tabel dalam database
- field: nama field dalam tabel
- field1: field where
- value1: value untuk field where (field1)
- order: 0=asc, 1=desc
- default: nilai kembalian default

contoh isi tabel 'lookup':

a	**b**	**c**
a1	b1	c1
a2	b2	c2

contoh:
sqliteboy_lookup2('lookup', 'c', 'a', 'a1', 0, ':(')
-> 'c1'

```
sqliteboy_lookup2('lookup', 'c_notfound', 'a',
'a1', 0, ':(')
-> ':('

sqliteboy_lookup2('lookup', 'b', 'a', 'a1', 0, ':
(')
-> 'b1'

sqliteboy_lookup2(12345, 'b', 'a', 'a1', 0, ':(')
-> ':('
```

sqliteboy_lookup3(table, field, field1, value1, field2, value2, order, default)
Mencari ke dalam tabel dan melakukan query sesuai pola berikut:

SELECT <field> FROM <table> WHERE <field1>=<value1> and <field2>=<value2> ORDER BY rowid asc

atau

SELECT <field> FROM <table> WHERE <field1>=<value1> and <field2>=<value2> ORDER BY rowid desc

kemudian mengembalikan baris pertama (atau nilai kembalian default).

argumen:

- table: nama tabel dalam database
- field: nama field dalam tabel
- field1: field where
- value1: value untuk field where (field1)
- field2: field where (field2)

- value2: value untuk field where (field2)
- order: 0=asc, 1=desc
- default: nilai kembalian default

contoh isi tabel 'lookup':

a	**b**	**c**
a1	b1	c1
a2	b2	c2

contoh:
```
sqliteboy_lookup3('lookup', 'c', 'a', 'a1', 'b',
'b1', 0, ':(')
-> 'c1'

sqliteboy_lookup3('lookup', 'c', 'a', 'a1', 'b',
'b2', 0, ':(')
-> ':('

sqliteboy_lookup3(12345, 'c', 'a', 'a1', 'b', 'b1',
0, ':(')
-> ':('
```

sqliteboy_split1(s, separator, table, column, convert)

Memecah string berdasarkan separator, kemudian melakukan insert pada tabel pada kolom yang telah ditentukan, untuk setiap anggota dalam list (hasil pemecahan). Apabila diperlukan, konversi dapat dilakukan sesuai tipe kolom.

argumen:

- s: string input

- separator: string separator
- table: nama tabel dalam database, untuk insert
- column: nama kolom dalam tabel
- convert: 0=tidak dilakukan konversi, 1=konversi ke tipe kolom apabila memungkinkan (atau ke string)

nilai kembalian:

- jumlah baris yang diinsert ke dalam tabel, atau 0

contoh:

```
sqliteboy_split1('h.e.l.l.o.w.o.r.l.d', '.',
'test_split', 'c', 1)
-> 10

sqliteboy_split1('hello', '', 'test_split', 'c', 0)
-> 1
```

tips:

- apabila empty string digunakan sebagai separator, maka whitespace akan digunakan sebagai separator.

sqliteboy_list_datetime1(s, n, interval, table, column, local)

Menghasilkan sejumlah daftar datetime dimulai dari datetime awal s (termasuk), sebanyak n, sesuai interval, kemudian melakukan insert ke dalam tabel pada kolom yang telah ditentukan, untuk setiap anggota datetime dalam daftar yang dihasilkan. Kita dapat pula menentukan apakah akan mempergunakan waktu UTC atau lokal.

argumen:

- s: datetime awal (YYYY-MM-DD HH:MM:SS)
- n: banyaknya datetime yang ingin dihasilkan, harus > 0
- interval: interval/jeda waktu dalam detik, harus > 0
- table: nama tabel dalam database, untuk insert
- column: nama kolom dalam tabel
- local: 0=UTC, 1=waktu lokal

nilai kembalian:

- jumlah baris yang diinsert ke dalam tabel, atau 0

```
contoh (timezone adalah UTC+7):
sqliteboy_list_datetime1('', 5, 60*60*24,
'test_date', 'a', 1)
-> 5
(data dalam table)
2013-06-03 23:13:27
2013-06-04 23:13:27
2013-06-05 23:13:27
2013-06-06 23:13:27
2013-06-07 23:13:27

sqliteboy_list_datetime1('', 5, 60*60*24,
'test_date', 'a', 0)
-> 5
(data dalam table)
2013-06-03 16:14:09
2013-06-04 16:14:09
2013-06-05 16:14:09
2013-06-06 16:14:09
```

```
2013-06-07 16:14:09

sqliteboy_list_datetime1('', 5, -60*60*24,
'test_date', 'a', 1)
-> 5
(data dalam table)
2013-06-03 23:14:55
2013-06-02 23:14:55
2013-06-01 23:14:55
2013-05-31 23:14:55
2013-05-30 23:14:55

sqliteboy_list_datetime1('2013-01-01 00:00:00', 5,
60*60, 'test_date', 'a', 1)
-> 5
(data dalam table)
2013-01-01 00:00:00
2013-01-01 01:00:00
2013-01-01 02:00:00
2013-01-01 03:00:00
2013-01-01 04:00:00
```

tips:

- apabila waktu awal merupakan string kosong, maka datetime saat ini akan digunakan (lokal).

sqliteboy_http_remote_addr()
Mendapatkan alamat IP remote (pengunjung).

sqliteboy_http_user_agent()
Mendapatkan user agent pengunjung. Pada umumnya, ini merupakan string identifikasi web browser yang digunakan.

sqliteboy_app_title()
Mendapatkan titel aplikasi SQLiteBoy.

contoh:
```
sqliteboy_app_title()
-> 'sqliteboy 1.44'
```

sqliteboy_var_set(name, value)
Pada user-defined variable, fungsi ini digunakan untuk mengassign nilai ke variabel, sesuai jumlah variabel yang diijinkan per user.

argumen:
- name: nama variabel, hanya menerima underscore dan/atau alfanumerik, tidak case-sensitive.
- value: nilai/isi variabel

nilai kembalian:
- 1 (berhasil), 0 (gagal)

contoh:
```
sqliteboy_var_set('a', 1000)
-> 1

sqliteboy_var_set('b', 'hello')
-> 1
```

tips:
- mengassign empty string atau 0 ke suatu variabel tidak akan menghapus variabel tersebut.
- untuk menghapus variabel serta menghemat memory, gunakanlah fungsi sqliteboy_var_del() yang juga

akan kita bahas.

sqliteboy_var_get(name)
Pada user-defined variable, fungsi ini digunakan untuk mendapatkan nilai dari variabel.

argumen:
- name: nama variabel, hanya menerima underscore dan/atau alfanumerik, tidak case-sensitive.

nilai kembalian:
- nilai/isi variabel atau ''

contoh:
```
sqliteboy_var_get('a')
-> 1000

sqliteboy_var_get('b')
-> hello
```

sqliteboy_var_del(name)
Pada user-defined variable, fungsi ini digunakan untuk menghapus variabel.

argumen:
- name: nama variabel, hanya menerima underscore dan/atau alfanumerik, tidak case-sensitive.

nilai kembalian:
- 1 (berhasil), 0 (gagal)

contoh:
```
sqliteboy_var_del('a')
```

```
-> 1

sqliteboy_var_del('b')
-> 1
```

sqliteboy_strip_html(s)
Mengabaikan tag HTML dan hanya mengambil isinya.

argumen:
- s: string input berupa kode HTML

```
contoh:
sqliteboy_strip_html('<b>hello</b>')
-> 'hello'
```

sqliteboy_x_user()
Mendapatkan nama user yang sedang login.

nilai kembalian:
- nama user apabila extended feature diaktifkan, atau ''

sqliteboy_x_profile_all(u, field, system)
Membaca profile user (system ataupun user-defined).

argumen:
- u: nama user
- field: field profile
- system: 0=user-defined, 1=system

nilai kembalian:

- isi dari field apabila extended feature diaktifkan dan field telah diset, atau ''

sqliteboy_x_profile(u, field)
Pada user-defined profile, fungsi ini digunakan untuk mendapatkan isi field custom, untuk user tertentu.

Kita akan membahas referensi user-defined profile, di bagian lain dalam buku ini.

argumen:
- u: nama user
- field: field dalam user-defined profile

nilai kembalian:
- isi dari field apabila extended feature diaktifkan dan field telah diset, atau ''

sqliteboy_x_profile_system(u, field)
Mendapatkan isi field dari system profile, untuk user tertentu.

Kita akan membahas referensi system profile, di bagian lain dalam buku ini.

argumen:
- u: nama user
- field: field profile

nilai kembalian:
- isi dari field apabila extended feature

diaktifkan dan field telah diset, atau ''

sqliteboy_x_my(field)
alias dari sqliteboy_x_profile(sqliteboy_x_user(), field)

sqliteboy_x_my_system(field)
alias dari
sqliteboy_x_profile_system(sqliteboy_x_user(), field)

7. Referensi: Number To Words

Di bagian ini, kita membahas referensi fungsi terbilang sqliteboy_number_to_words(s, language).

Fungsi ini mendukung berbagai bahasa dan berikut adalah daftar bahasa yang didukung. Pemanggilan fungsi menggunakan kode bahasa.

Kode bahasa	Bahasa
id	Bahasa Indonesia
en1	Bahasa Inggris, dengan skema trillion billion million thousand

Saat ini, fungsi mendukung sampai nama bilangan besar triliun (10 pangkat 12 atau 1.000.000.000.000), sehingga mendukung sampai triliun triliun.

Pemisah ribuan yang digunakan dalam referensi ini adalah titik dan pemisah desimal adalah koma, karena buku ini ditulis dalam Bahasa Indonesia.

Batasan bilangan yang didukung dimulai dari:
-999.999.999.999.999.999.999.999.999,99...
(min 999,99999999999999999999999 triliun triliun)
(ditambah angka di belakang koma)

sampai:
999.999.999.999.999.999.999.999.999,99...
(999,99999999999999999999999 triliun triliun)

(ditambah angka di belakang koma)

Walaupun, bisa saja, batasan ini akan berbeda untuk bahasa tertentu.

Untuk angka di belakang koma, sesungguhnya hanya dibatasi oleh batasan tipe float python, yang merupakan angka yang sangat-sangat-sangat besar. Dengan demikian, contoh berikut merupakan bilangan yang valid dan didukung oleh fungsi:

```
999999999999999999999999999,99999999999999999999999999999999999999999999999999
```

Yang sangat penting, untuk bilangan yang sangat besar, kita perlu melewatkan ke dalam fungsi sebagai string, dan bukan bilangan. Tentu saja, kita selalu bisa melewatkan sebagai string untuk bilangan kecil ataupun besar.

Untuk contoh penggunaan fungsi, bacalah juga referensi UDF.

8. Referensi: Form

Kita telah membahas fitur form pada topik **Mengenal SQLiteBoy**, di awal buku ini.

Di bagian ini, kita membahas referensi kode form.

Pastikanlah extended feature telah diaktifkan dan user yang login adalah level admin, sehingga form dapat dibuat dan/atau diedit.

Setiap form dapat diatur untuk hanya boleh dijalankan oleh user-user tertentu saja, atau oleh semua user.

Catatan penting:

- Terminologi yang benar dalam definisi/sintaks form harusnya menuruti istilah JSON.
- Akan tetapi, demi memudahkan developer yang terbiasa menggunakan python namun mungkin tidak terlalu terbiasa dengan JSON, kita akan menggunakan istilah yang setara, dalam python.

Sintaks

- Form didefinisikan dengan kode JSON. Setiap kali kode form disimpan, maka pemeriksaan sintaks akan dilakukan, dan pesan kesalahan yang sesuai akan ditampilkan.
- Pengguna python umumnya akan familiar dengan kode JSON karena sangat mirip dengan dictionary.

Tapi, pastikanlah bahwa setiap string (termasuk key) harus dikutip ganda (antara " dan ") dan koma di akhir dictionary atau list tidak diijinkan.

- Setiap key adalah case-sensitive, di mana huruf besar/kecil dibedakan.
- Setiap definisi form adalah dict ({}).

Database

- Form hanya dapat bekerja dengan satu tabel, dan operasi default adalah insert (walaupun dapat dicegah).
- Bagaimana kalau kita perlu menulis ke berbagai tabel, melakukan operasi selain insert atau menjalankan banyak SQL Query yang kompleks? Kita bisa selalu gunakan berbagai statemen SQL Query tambahan, yang dapat diatur agar dijalankan (sesuai urutan definisi) sebelum dan/atau sesudah form diproses.
- Setiap field pada form adalah field pada tabel. Kita tidak bisa mendefinisikan field pada form tanpa keberadaan field tersebut pada tabel.
- Relasi antar tabel tidak harus dimaintain pada level database (walau tidak dilarang). Sebagai alternatif, kita bisa gunakan fitur reference, contraint, default, event onsave atau lainnya yang disediakan oleh SQLiteBoy, untuk membantu diantaranya menjaga integritas data.

Subform

- Form mendukung subform sederhana. Subform sendiri hanya memiliki sebagian dari fitur form dan subform tidak dapat berdiri sendiri.
- Hanya reference dan default yang didukung dalam subform. Semua field wajib diisi dan tidak ada yang readonly. Kolom-kolom dalam tabel yang ingin ditampilkan dalam subform tetap dapat dipilih.
- Subform umumnya digunakan dalam relasi one-to-many. Sebagai contoh, dalam form penjualan, kita mencatat nama pelanggan dalam form dan item-item yang dibeli dalam subform (satu pelanggan bisa membeli banyak item).
- Dari sisi user interface, form yang dilengkapi dengan subform ditampilkan pada layar yang sama.

Message

- Setiap kali form diproses, apapun hasilnya, message default akan ditampilkan. Tentu saja, kita bisa mengubah message tersebut sesuai dengan aplikasi yang dibangun.
- Message mendukung kode HTML, sehingga kita bisa membuat hyperlink ke form/report lain, sebagai contoh.
- Message pada form dapat ditentukan sesuai nilai kembalian dari query. Antara gagal atau sukses, sebagai contoh, message bisa berbeda.

Mencegah insert

- Sebagaimana dibahas sebelumnya, operasi default adalah insert. Jadi, pada saat form disubmit oleh user, apabila kita tidak mencegah operasi insert, maka insert selalu dilakukan ke tabel yang didefinisikan dalam form.
- Hal tersebut sebenarnya tidak masalah, karena apabila ingin menulis ke tabel lain ataupun melakukan tindakan lain, kita bisa gunakan berbagai statemen SQL Query tambahan.
- Apabila kita benar-benar ingin tidak ada operasi insert yang dilakukan ke tabel, maka aturlah key 'insert' pada definisi form menjadi nilai 0 atau lebih kecil. Efek dari ini adalah nilai last insert rowid dan hasil query akan disamakan dengan nilai yang diset untuk key 'insert' tersebut.

Python handler

- Setiap form dapat memiliki satu python handler.
- Kita akan membahas tentang python handler di bagian lain, di dalam buku ini.

Primary key

- Apabila terdapat kolom primary key dalam data form, maka secara default, penanda berupa karakter * akan ditambahkan pada label kolom.

Key

Perhatikanlah bahwa form didefinisikan dengan sebuah dict. Apa yang kita bahas adalah key dari dict tersebut.

data

Tipe	list dari dict
Status	Diperlukan

Merupakan data dari form. Setiap field yang ingin ditampilkan pada form didefinisikan dalam data. Kita akan membahas tersendiri semua key yang didukung pada bagian key (data).

security

Tipe	dict
Status	Diperlukan

Merupakan pengaturan keamanan form. Siapa saja yang boleh menjalankan form didefinisikan dalam security. Kita akan membahas tersendiri semua key yang didukung pada bagian key (security).

title

Tipe	str
Status	Opsional
Contoh	"My Form"

Merupakan judul dari form.

info

Tipe	str
Status	Opsional
Contoh	"Form Information"
HTML	Diijinkan

Merupakan informasi form.

sub

Tipe	list
Status	Opsional

Merupakan definisi subform. Harus berupa list yang terdiri dari 5 anggota:

- tabel subform (str)

- kolom relasi dalam tabel subform (str), ke tabel form. Perhatikanlah bahwa nilai kembalian dari fungsi SQLite last_insert_rowid() hasil insert dari table form akan dituliskan ke dalam kolom ini untuk setiap baris input subform. Tergantung desain tabel form, kita bisa mendapatkan relasi dengan membaca kolom ROWID apabila diperlukan.
- list dari [jumlah baris (int), jumlah baris wajib diisi (int)]
- list dari list kolom [kolom (str), label (str), reference, default]. Untuk reference dan default, kita akan membahasnya dalam Key (data).
- informasi subform (str)

contoh:

```
["table2", "a", [5,3], [["b", "Column B", [ ["0", "NO"], ["1", "YES"] ], "1"], ["c", "Column C", "select a, b from table1", ""]], "My Subform"]
```

Dalam contoh tersebut:

- subform akan mempergunakan table2.
- nilai kembalian dari fungsi last_insert_rowid() hasil insert dari table form akan dituliskan ke dalam kolom a.
- subform tersebut akan terdiri dari 5 baris, di mana 3 baris wajib diisi.
- Subform tersebut akan menampilkan dua kolom dari table2, yaitu b dan c. Kolom b akan ditampilkan

dengan label Column B dan kolom c akan ditampilkan dengan label Column C.

- judul atau informasi dari subform adalah My Subform.

message

Tipe	list
Status	Opsional
HTML	Diijinkan

Merupakan pengaturan message setelah form diproses. Hanya berlaku untuk form (tidak berlaku untuk subform).

Harus berupa list yang terdiri dari 3 anggota:

- message apabila hasil query mengembalikan < 0 (str)
- message apabila hasil query mengembalikan 0 (str)
- message apabila hasil query mengembalikan > 0 (str)

Beberapa variabel berikut tersedia dan dapat digunakan dalam message:

- $result akan digantikan dengan nilai hasil query

- $<column> akan digantikan dengan nilai input user untuk <column>. Sebagai contoh, apabila terdapat kolom nama yang ditampilkan dalam form, maka kita bisa menggunakan $nama dalam message, yang akan digantikan dengan apa yang diinput oleh user pada form.
- $last_insert_rowid akan digantikan dengan nilai kembalian dari fungsi last_insert_rowid().
- $python_handler akan digantikan dengan nilai kembalian dari python handler apabila disediakan. Default adalah -1.

contoh:

```
["unknown result", "zero result", "success: $result"]
```

sql2

Tipe	list
Status	Opsional

Merupakan statement SQL Query tambahan, yang akan dikerjakan setelah form diproses.

Harus merupakan list dari str, dimana setiap anggota list adalah SQL Query yang akan dikerjakan. Urutan adalah penting karena query akan dikerjakan sesuai urutan definisi.

Beberapa variabel tersebut tersedia dan dapat digunakan dalam query:

- $<column> akan digantikan dengan nilai input user untuk <column>. Sebagai contoh, apabila terdapat kolom nama yang ditampilkan dalam form, maka kita bisa menggunakan $nama dalam query, yang akan digantikan dengan apa yang diinput oleh user pada form.
- $last_insert_rowid akan digantikan dengan nilai kembalian dari fungsi last_insert_rowid() setelah penulisan ke table form. (hanya berlaku untuk key sql2 dan bukan sql0 yang akan dibahas berikutnya).
- Setiap statemen akan dikerjakan dalam transaksi.
- Satu hal yang sangat penting adalah quoting akan dilakukan otomatis. Dengan demikian, kita tidak perlu menuliskan tanda kutip dalam query, walaupun untuk kolom dengan tipe teks. Lihatlah juga pada contoh berikut.

contoh:

```
["insert into table3(a, b, c, d, e) values($a, $b, $c, $d, $e)", "insert into table4(x) values($last_insert_rowid)"]
```

Bisa kita lihat bahwa terdapat 2 query yang didefinisikan.

Perhatikanlah query pertama. Kita anggap bahwa dalam form terdapat 5 kolom, yaitu a, b, c d dan e, yang semuanya bertipe varchar.

Tanpa quoting dilakukan otomatis, kita perlu menuliskan ...values("$a",....Hal tersebut tidaklah diperlukan karena quoting akan ditambahkan dengan sendirinya.

sql0

Tipe	list
Status	Opsional

Merupakan statement SQL Query tambahan, yang akan dikerjakan sebelum form diproses.

Lihatlah juga pembahasan tentang sql2 apabila diperlukan.

insert

Tipe	int
Status	Opsional

Sebagaimana telah dibahas sebelumnya, insert dapat digunakan untuk mencegah operasi insert ke dalam tabel form, apabila nilai 0 atau lebih kecil diberikan.

confirm

Tipe	str
Status	Opsional

Apabila diisi, maka dialog konfirmasi (javascript) akan ditampilkan ketika form disubmit. Hal ini dapat membantu mencegah submit yang dilakukan tanpa sengaja.

Key (data)

Perhatikanlah bahwa data merupakan list dari dict. Apa yang kita bahas adalah key dari dict dalam list. Menentukan kolom-kolom apa saja yang akan ditampilkan dalam form, berikut properti dari kolom tersebut. Urutan definisi kolom akan menentukan urutan/posisi dalam form ketika dijalankan.

table

Tipe	str
Status	Diperlukan

Contoh	"table1"

Merupakan nama tabel yang digunakan dalam form. Hanya satu tabel yang didukung.

Apabila beberapa nama tabel diberikan untuk setiap anggota list, maka hanya yang pertama yang akan dianggap, dan selebihnya akan diabaikan.

column

Tipe	str
Status	Diperlukan
Contoh	"col1"

Merupakan nama kolom dalam tabel, yang akan ditampilkan dalam form.

label

Tipe	str
Status	Opsional
Contoh	"column 1"

Merupakan label untuk kolom dalam tabel.

required

Tipe	int
Status	Opsional
Contoh	1

Menentukan apakah kolom tersebut wajib diisi atau tidak:

- Apabila diisikan 0, maka kolom tersebut opsional dan form bisa diproses tanpa kolom tersebut perlu diisi.
- Apabila diisikan 1, maka kolom tersebut wajib diisi oleh user.

readonly

Tipe	int
Status	Opsional
Contoh	0

Menentukan apakah suatu kolom readonly atau tidak:

- Apabila diisikan 0, maka kolom tersebut dapat diisi/diubah oleh user.
- Apabila diisikan 1, maka kolom tersebut tidak

dapat diisi/diubah oleh user.

reference

Tipe	str, list, atau int
Status	Opsional

Menentukan nilai-nilai yang dapat dipilih oleh user, dimana nilai-nilai tersebut didapatkan dari list (statik) ataupun hasil query. Selain itu, reference juga dapat digunakan untuk menentukan flag input.

Apabila reference diberikan nilai dengan tipe str, maka:

- hasil query akan digunakan dan nilai yang diberikan dalam reference dianggap sebagai SQL Query.
- SQL Query diharapkan untuk mengembalikan dua kolom: a dan b. Kolom a berisikan nilai sesungguhnya (atau id) dan kolom b dapat berisikan label.
- Pada user interface SQLiteBoy, HTML Select akan digunakan, dimana value dari select akan diambil dari kolom a.

Apabila reference diberikan nilai dengan tipe list, maka:

- nilai statik akan digunakan
- list diharapkan berisikan anggota-anggota berupa list, yang masing-masing memiliki dua anggota.
- Anggota pertama dalam setiap list berisikan nilai sesungguhnya (atau id) dan anggota kedua dalam setiap list dapat berisikan label.
- Pada user interface SQLiteBoy, HTML Select akan digunakan, dimana value dari select akan diambil dari anggota pertama setiap list.

Apabila reference diberikan nilai dengan tipe int, maka:

- akan dianggap sebagai flag input
- apabila nilai yang diberikan adalah 2, maka pada user interface SQLiteBoy, HTML input bertipe password akan digunakan.

contoh:

- nilai bertipe str (SQL Query):

"select col1 as a, col2 as b from table1"

- nilai bertipe list:

[["0", "NO"], ["1", "YES"]]

- nilai bertipe int

2

default

Tipe	str, list atau int
Status	Opsional

Menentukan nilai default.

Apabila default diberikan nilai dengan tipe str atau int maka:

- akan digunakan apa adanya.

Apabila default diberikan nilai dengan tipe list, dan anggota pertama dalam list bukan string kosong, maka:

- akan dianggap sebagai pemanggilan fungsi SQL.
- list harus berisikan paling sedikit satu anggota bertipe str, yang akan dianggap sebagai nama fungsi.
- Apabila list berisikan lebih dari satu anggota,

maka anggota kedua dan seterusnya (apabila ada) akan dilewatkan sebagai parameter ketika pemanggilan fungsi dilakukan.

- Nilai kembalian dari pemanggilan fungsi akan digunakan sebagai nilai default.
- Pola: [nama_fungsi, arg1,...]
- Ketika menuliskan nama fungsi, harap untuk tidak menambahkan ().

Apabila default diberikan nilai dengan tipe list, dengan dua anggota, maka:

- akan dianggap sebagai SQL Query.
- kedua anggota harus bertipe str. Anggota pertama harus berupa string kosong dan anggota kedua berisikan SQL Query.
- SQL Query harus mengembalikan satu kolom, yaitu a, yang nilainya akan digunakan sebagai nilai default.

Contoh:

```
["sqliteboy_md5", "hello"]
```

```
["sqlite_version"]
```

constraint

Tipe	list
Status	Opsional

Menentukan nilai yang dianggap valid, sebelum form diproses.

Harus merupakan list dengan 4 anggota:

["nama_fungsi", as_str, "kondisi", "pesan_kesalahan"]

catatan:

- nama_fungsi dapat dikosongkan.
- as_str harus bernilai 1 atau 0. Apabila bernilai 1, maka argumen akan dilewatkan sebagai string pada saat pemanggilan fungsi.
- kondisi tidak boleh dikosongkan dan harus berisikan perbandingan boolean.
- pesan_kesalahan dapat dikosongkan.

Apabila nama_fungsi tidak kosong:

- nama_fungsi akan dipanggil dengan input user untuk kolom tersebut sebagai argumen.
- Nilai kembalian dari fungsi akan dibandingkan dengan kondisi.

Apabila nama_fungsi kosong:

- input user untuk kolom tersebut akan dibandingkan dengan kondisi.

Apabila hasil perbandingan adalah 0 (false), maka form tidak akan diproses lebih lanjut. Apabila pesan_kesalahan tidak kosong, maka pesan_kesalahan tersebut akan ditampilkan. Apabila pesan_kesalahan kosong, maka pesan kesalahan generik dengan nama kolom, nama_fungsi (apabila ada) dan kondisi akan ditampilkan.

contoh:

```
["", 0, "> 10", "must be larger than 10"]
```

input user akan dianggap valid untuk kolom tersebut apabila perbandingan dengan: > 10 adalah benar. Bisa kita lihat, dalam contoh ini, tidak ada pemanggilan fungsi yang dilakukan.

```
["sqliteboy_len", 1, "> 10", ""]
```

input user akan dilewatkan ke fungsi sqliteboy_len sebagai string dan hasil pemanggilan fungsi tersebut akan dibandingkan dengan: > 10. Input user akan dianggap valid apabila perbandingan benar.

tips:

- fungsi sqliteboy_as_integer() dapat digunakan untuk konversi input user ke integer sebelum perbandingan dilakukan.
- Input dari user akan selalu dilewatkan sebagai string. Lakukanlah konversi apabila diperlukan.

onsave

Tipe	str
Status	Opsional

Apabila sebuah kolom memiliki event onsave, maka SQL Query yang ditentukan akan dipanggil terlebih dahulu dan barulah nilai kembalian dari query tersebut yang akan digunakan. Jadi, tidak langsung dari input user.

catatan:

- SQL Query yang diberikan dapat sangat kompleks dan melibatkan pemanggilan fungsi bertingkat.
- SQL Query harus mengembalikan satu kolom yaitu onsave.
- Quoting akan dilakukan secara otomatis.
- $value akan digantikan dengan input dari user.

contoh:

```
"select $value || ' : ' || sqliteboy_upper( sqliteboy_md5($value)) as onsave"
```

pada contoh tersebut, MD5 hash dari input user akan dihitung dengan fungsi sqliteboy_md5(). Setelah itu, nilai kembalian fungsi akan diubah ke huruf besar dengan fungsi sqliteboy_upper(). Setelah itu, nilai kembalian fungsi akan digabungkan dengan string lainnya (final).

Untuk contoh tersebut, dengan input user berupa string hello, maka event onsave akan menghasilkan nilai:

```
hello : 5D41402ABC4B2A76B9719D911017C592
```

Key (security)

Perhatikanlah bahwa security merupakan dict. Apa yang kita bahas adalah key dari dict tersebut. Menentukan pengaturan keamanan form.

run

Tipe	list atau string kosong
Status	Diperlukan

Menentukan user level standard mana saja yang dapat menjalankan form. User dengan level admin selalu dapat menjalankan form.

Apabila diberikan nilai berupa string kosong, maka semua user dapat menjalankan form tersebut.

Apabila diberikan nilai dengan tipe list, maka hanya user-user yang terdapat dalam list yang dapat menjalankan form tersebut.

contoh:

- []
- ["user1", "user2"]

Contoh kode form 1

```
{
  "title" : "My Form (Simple)",
  "info"  : "Form Information",
  "data"  : [
              {
                "table"     : "table1",
                "column"    : "a"
              },
```

```
              {
                "table"     : "table1",
                "column"    : "d"
              },
              {
                "table"     : "table1",
                "column"    : "f"
              }
            ],
  "security" : {
                 "run" : ""
               }
}
```

Contoh kode form 2

```
{
  "title" : "My Form 1",
  "info"  : "Form Information",
  "sub"   : [
              "table2",
              "a",
              [5,3],
```

```
              [
                ["b", "Column B", [ ["0", "NO"],
["1", "YES"] ], "1"],
                ["c", "Column C", "select a, b from
table1", ""]
              ],
              "My Subform"
            ],
  "sql2"  : [
              "insert into table3(a, b, c, d, e)
values($a, $b, $c, $d, $e)",
              "insert into table4(x)
values($last_insert_rowid)"
            ],
  "data"  : [
              {
                "table"     : "table1",
                "column"    : "a",
                "label"     : "column a",
                "required"  : 1,
                "reference" : [ ["0", "NO"], ["1",
"YES"] ],
                "default"   : "1"
              },
              {
```

```
                "table"     : "table1",
                "column"    : "b",
                "reference" : "select
sqliteboy_randrange(1, 100000000000) as a, 'hello '
|| sqliteboy_time() as b from _sqliteboy_"
              },
              {
                "table"     : "table1",
                "column"    : "c",
                "default"   : ["sqliteboy_md5",
"hello"],
                "constraint": ["sqliteboy_len", 1,
"= 32", ""],
                "onsave"    : "select
sqliteboy_upper($value) as onsave"
              },
              {
                "table"     : "table1",
                "column"    : "d",
                "label"     : "d (incorrect larger
than 100)",
                "required"  : 1,
                "constraint": ["", 0, "> 100",
"must be larger than 100"]
              },
              {
```

```
                "table"     : "table1",
                "column"    : "e",
                "label"     : "e (correct larger
than 100)",
                "required"  : 1,
                "constraint":
["sqliteboy_as_integer", 1, "> 100", "must be
larger than 100"]
              },
              {
                "table"     : "table1",
                "column"    : "f"
              }
            ],
  "message"  : ["unknown result", "zero result",
"success: $result"],
  "security" : {
                 "run" : ""
               }
}
```

9. Referensi: Report

Kita telah membahas fitur report pada topik **Mengenal SQLiteBoy**, di awal buku ini.

Di bagian ini, kita membahas referensi kode report.

Pastikanlah extended feature telah diaktifkan dan user yang login adalah level admin, sehingga report dapat dibuat dan/atau diedit.

Setiap report dapat diatur untuk hanya boleh dijalankan oleh user-user tertentu saja, atau oleh semua user.

Mirip dengan form

- Form dan Report memiliki cukup banyak kemiripan antar keduanya.
- Hal ini termasuk sintaks yang digunakan.
- Terminologi tipe data yang digunakan dalam pembahasan ini juga sama dengan form.
- Report juga menggunakan sebagian besar aturan dan fitur field milik form. Apabila ada perbedaan, kita akan membahasnya.
- Apabila diperlukan, bacalah juga referensi form.

Parameter dan hasil report

- Sebagaimana dibahas di awal buku ini, ketika suatu report dijalankan, kita memiliki kesempatan untuk memasukkan berbagai parameter yang dapat menentukan hasil report.
- Di SQLiteBoy, parameter bisa nol atau lebih, namun tahapan untuk memasukkan parameter harus selalu dilalui.
- Sebagai contoh, pada report dengan 2 parameter, user memasukkan kedua parameter, kemudian melanjutkan ke tahap hasil report dengan klik tombol.
- Contoh lainnya, pada report tanpa parameter, user tidak perlu memasukkan apa-apa, dan layar tersebut hanya berisikan tombol untuk melanjutkan ke tahap hasil report. Namun layar tersebut tetap harus dilalui.
- Bahwa layar tersebut harus tetap dilalui diantaranya karena di layar tersebut, kita bisa menentukan format report.
- Parameter yang didefinisikan harus selalu diisi. Ini berbeda dengan form dimana kita bisa menentukan kolom mana saja yang required. Di report, semua parameter adalah required. Gunakanlah nilai default agar user tidak harus selalu mengisikan nilainya.
- Dalam buku ini ataupun situs web SQLiteBoy, kita mungkin/juga mempergunakan istilah search atau wizard untuk layar di mana kita memasukkan berbagai parameter.

Format report

- HTML lengkap dengan semua menu (default).
- HTML tanpa menu dan warna (printer friendly). Berguna apabila report tersebut ingin dicetak langsung dari web browser.
- CSV, berguna apabila report tersebut ingin diolah lagi, barangkali secara manual, sebagai contoh mempergunakan program spreadsheet.
- PDF, berguna apabila report ingin disimpan dan dicetak (lagi) di lain waktu (barangkali di sistem lain). Kita dapat menentukan ukuran kertas dan margin. Perhatikanlah bahwa fitur ini tidak selalu tersedia, sesuai ketersediaan modul yang diperlukan di sistem.

SQL Query dan hasil report

- SQLiteBoy tidak membatasi jenis query yang diterima dalam definisi report (select, insert, update, delete, atau lainnya).
- SQLiteBoy akan mendeteksi hasil kembalian dari query. Berupa data tabular ataupun bilangan, SQLiteBoy akan menyesuaikan dengan aturan yang telah didefinisikan dalam report.
- Hasil dari SQL Query adalah isi report, kecuali python handler tersedia untuk report tersebut, yang mana akan menggantikan query.
- Untuk hasil query atau kembalian python handler berupa data tabular, maka urutan kolom akan ditampilkan sesuai definisi report.

Header dan footer

- Header dan footer didukung dan selalu ditampilkan berupa tabel. Satu untuk header, satu untuk footer.
- Selalu ada header dan footer default apabila kita tidak mendefinisikan header dan footer sendiri. Jadi, definisikanlah header atau footer, apabila kita tidak ingin menggunakan yang default.
- Jumlah kolom dalam tabel header atau footer diambil dari kolom terbanyak yang didefinisikan untuk setiap barisnya.
- Di dalam setiap sel tabel, kita bisa mengisikan teks biasa, hasil dari query ataupun berupa gambar. Sebagai catatan, gambar tidak didukung pada format CSV.

Header default

<Judul_report>	<Info_report>
[Key 1]	[input 1]
[…]	[…]
[Key n]	[input n]

Footer default (select)

<Jumlah_baris>	Tulisan "row(s)" ataupun terjemahannya

Footer default (non-select)

<message> atau kosong	

Parameter dan tabel database

- Berbeda dengan form, report tidak bekerja dengan tabel database.
- Apabila pada form, kolom yang tampil merupakan kolom dalam tabel, maka tidak demikian dengan report.
- Parameter dalam report ditentukan dengan sebuah nama key. Nantinya, key tersebut akan ditampilkan sebagai HTML input. Dan, apapun yang diisikan oleh user, dapat diakses dari SQL Query ataupun python handler.

Message

- Sebagaimana pada form, report juga mendukung message berdasarkan hasil dari query.
- Tapi, ini hanya berlaku apabila query mengembalikan bilangan, dan bukan berupa data

tabular.
- Bacalah pembahasan tentang message pada form, apabila diperlukan.

Python handler

- Setiap report dapat memiliki satu python handler.
- Kita akan membahas tentang python handler di bagian lain, di dalam buku ini.

Key

Perhatikanlah bahwa report didefinisikan dengan sebuah dict. Apa yang kita bahas adalah key dari dict tersebut.

data

Tipe	list dari dict
Status	Diperlukan, walaupun dapat berupa list tanpa anggota (kosong).

Merupakan data wizard/search dari report, tempat kita mendefinisikan parameter. Kita akan membahas tersendiri semua key yang didukung pada bagian key (data).

security

Tipe	dict
Status	Diperlukan

Merupakan pengaturan keamanan report. Siapa saja yang boleh menjalankan report didefinisikan dalam security. Kita akan membahas tersendiri semua key yang didukung pada bagian key (security).

sql

Tipe	str
Status	Diperlukan

Merupakan SQL Query untuk menentukan isi report.

Untuk mengakses nilai yang diinput oleh user dalam parameter report (apabila didefinisikan), kita menggunakan placeholder berupa $ diikuti oleh nama key.

Quoting akan dilakukan secara otomatis apabila diperlukan.

contoh:

```
"select a.a as 'column a of table1', a.e from table1 a where a.a = $input_a_a and a.e > $a_e"
```

Untuk contoh tersebut, berarti kita harus memiliki setidaknya dua nama key, yaitu input_a_a dan a_e, didefinisikan dalam data.

tips:

- Karena sql selalu harus diisi, maka kita bisa menggunakan nilai seperti "select 1", apabila kita tidak memerlukan hasil query, karena report akan dihasilkan dari python handler.

title

Sama seperti pada form.

info

Sama seperti pada form.

header

Tipe	list
Status	Opsional

Menentukan kolom-kolom apa saja yang akan ditampilkan, beserta urutan dari kiri ke kanan, untuk report berupa data tabular.

Apabila tidak ditentukan, maka semua kolom dalam

hasil report akan ditampilkan, dalam urutan kolom yang tidak terprediksi.

contoh:

```
["column a of table1", "e"]
```

align

Tipe	list
Status	Opsional

Menentukan apakah teks dalam kolom diformat rata kiri, tengah, kanan atau kiri/kanan. Harus merupakan list dengan semua anggota bertipe int.

Hanya berlaku apabila:

- Format report adalah HTML (termasuk versi printer friendly) atau PDF.
- Key header telah ditentukan.

Untuk setiap kolom dalam header, maka nilai-nilai berikut dapat digunakan untuk align.

0	Rata kiri
1	Rata tengah
2	Rata kanan
3	Rata kiri/kanan

contoh:

```
[1, 2]
```

Dalam contoh ini, kolom pertama dalam header akan diformat rata tengah dan kolom kedua akan diformat rata kanan.

message

Tipe	list
Status	Opsional
HTML	Diijinkan

Merupakan pengaturan message khusus untuk report yang mengembalikan nilai berupa bilangan (bukan data tabular). Mirip dengan message pada form walaupun lebih terbatas.

Harus berupa list yang terdiri dari 3 anggota. Kita gunakan istilah nilai berikut ini, karena hasil report tidak harus selalu dari query, melainkan bisa dari python handler (ini berbeda dengan form):

- message apabila nilai < 0 (str)
- message apabila nilai 0 (str)
- message apabila nilai > 0 (str)

Beberapa variabel berikut tersedia dan dapat digunakan dalam message. Lebih terbatas dari message pada form:

- $result akan digantikan dengan nilai report

- $<key> akan digantikan dengan nilai input user untuk <key>. Sebagai contoh, apabila terdapat key nama yang ditampilkan dalam wizard report, maka kita bisa menggunakan $nama dalam message, yang akan digantikan dengan apa yang diinput oleh user pada wizard.

contoh:

```
["unknown result", "zero result", "success: $result"]
```

headers

Tipe	List dari list dari list
Status	Opsional
Contoh	Lihatlah pada contoh kode report 2

Menentukan custom header sehingga header default report tidak digunakan.

Harus merupakan list dari list (baris) dari list (kolom) yang memiliki 3 anggota, untuk setiap sel dalam tabel header.

Ketiga anggota tersebut:

- Anggota pertama (str) menentukan tipe sel: string kosong (teks apa adanya), sql (SQL Query) dan files.image (gambar dalam files).
- Anggota kedua (str atau int) menentukan nilai dari tipe sel.

- Anggota ketiga (dict) digunakan untuk atribut. Saat ini, dapat berupa dictionary kosong ({}).

Catatan untuk anggota kedua (nilai tipe sel), untuk tipe sel sql:

- $result_row_count akan digantikan dengan jumlah baris dalam report atau -1 apabila bukan merupakan data tabular.
- $result akan digantikan dengan kembalian dari query berupa bilangan bulat atau -1.
- $result_message akan digantikan dengan message report (atau string kosong untuk message custom).
- Setiap $<key> dalam data akan digantikan dengan input user untuk setiap key tersebut.
- Quoting akan dilakukan secara otomatis.
- Query harus mengembalikan satu kolom a.

Catatan untuk anggota kedua (nilai tipe sel), untuk tipe sel files.image:

- Nilai yang diterima adalah berupa bilangan yang merupakan id dari file, dalam files user.
- Pastikanlah file tersebut telah dishare, karena apabila tidak, maka hanya akan tampak dalam report apabila yang menjalankan report adalah pemilik file.

footers

lihatlah pembahasan tentang headers.

paper

Tipe	list
Status	Opsional

Menentukan ukuran kertas dalam satuan point (1/72 inci). Hanya berlaku untuk format report PDF.

Harus merupakan list dengan 2 anggota bertipe int/float yang masing-masing adalah lebar dan panjang kertas.

margins

Tipe	list
Status	Opsional

Menentukan ukuran margin dalam satuan point (1/72 inci). Hanya berlaku untuk format report PDF.

Harus merupakan list dengan 4 anggota bertipe int/float yang masing-masing adalah margin kiri, kanan, atas dan bawah.

confirm

Sama seperti pada form.

Key (data)

Perhatikanlah bahwa data merupakan list dari dict. Apa yang kita bahas adalah key dari dict dalam list. Menentukan parameter report. Secara umum sama dengan apa yang didukung pada form, walaupun lebih terbatas.

key

Tipe	str
Status	Diperlukan
Batasan	Underscore dan alfanumerik saja
Contoh	"input_a_a"

Merupakan key untuk parameter, yang nantinya akan ditampilkan berupa HTML input.

label

Tipe	str
Status	Opsional
Contoh	"column a ="

Menentukan label untuk key. Ada baiknya mencerminkan apa yang ingin kita lakukan dalam query atau python handler (lihatlah pada contoh).

readonly

Sama seperti pada form.

reference

Sama seperti pada form.

default

Sama seperti pada form.

constraint

Sama seperti pada form.

type

Tipe	str
Status	Opsional

Secara default, tipe dari input user adalah str.

Type digunakan untuk melakukan konversi tipe data input ke tipe data yang ditentukan.

Saat ini, hanya "integer" yang didukung, yang mana konversi akan dilakukan dengan int() milik python.

Key (security)

Sama seperti pada form.

Contoh kode report 1

```
{
  "title" : "My Report",
  "info"  : "Report Information",
  "header": ["column a of table1", "e"],
  "sql"   : "select a.a as 'column a of table1',
a.e from table1 a where a.a = $input_a_a and a.e >
$a_e",
  "data"  : [
              {
                "key"       : "input_a_a",
                "label"     : "column a equals",
                "reference" : [ ["0", "NO"], ["1",
"YES"] ],
```

```
                "default"   : "1"
              },
              {
                "key"       : "a_e",
                "label"     : "e (as integer) >",
                "constraint":
["sqliteboy_as_integer", 1, "> 0", "e must be
integer"]
              }
            ],
  "security" : {
                 "run" : ""
               }
}
```

Contoh kode report 2

```
{
  "title" : "My Report",
  "info"  : "Report Information",
  "header": ["column a of table1", "e"],
  "sql"   : "select a.a as 'column a of table1',
a.e from table1 a where a.a = $input_a_a and a.e >
$a_e",
  "data"  : [
```

```
              {
                "key"       : "input_a_a",
                "label"     : "column a equals",
                "reference" : [ ["0", "NO"], ["1",
"YES"] ],
                "default"   : "1"
              },
              {
                "key"       : "a_e",
                "label"     : "e (as integer) >",
                "constraint":
["sqliteboy_as_integer", 1, "> 0", "e must be
integer"]
              }
            ],
  "headers"  : [
                  [
                      ["files.image", "31", {}],
                      ["", "My Report", {}]
                  ],
                  [
                      ["", "Date/Time", {}],
                      ["sql", "select
sqliteboy_time3(sqliteboy_time()) as a", {}]
                  ],
```

```
                    [
                        ["", "User", {}],
                        ["sql", "select
sqliteboy_x_user() as a", {}]
                    ],
                    [
                        ["", "column a equals", {}],
                        ["sql", "select $input_a_a as
a", {}]
                    ],
                    [
                        ["", "e (as integer) >", {}],
                        ["sql", "select $a_e as a",
{}]
                    ],
                    [
                        ["", "Rows", {}],
                        ["sql", "select
$result_row_count as a", {}]
                    ]
                ],
  "security" : {
                  "run" : ""
                }
}
```

10. Referensi: Page

Kita telah membahas fitur page pada topik **Mengenal SQLiteBoy**, di awal buku ini.

Di bagian ini, kita membahas referensi kode page.

Pastikanlah extended feature telah diaktifkan.

Tipe	Kode	HTML
emphasis	~text~	text
strong	*text*	text
underline	_text_	<u>text</u>
link	[text|url]	text

Catatan:

- Page dapat diketikkan tanpa sintaks apapun. Apa yang kita bahas pada tabel di atas hanyalah untuk pemformatan teks sederhana.
- Setiap page akan ditampilkan dalam tag HTML <pre></pre>.
- Setiap tag yang dituliskan, apabila ada, tidak akan ikut disimpan dan/atau ditampilkan (hanya isinya saja).
- Perhatikanlah bahwa setiap user hanya dapat memiliki satu page.
- Apabila seorang user tidak membuat page untuk

dirinya, maka ketika URL page untuk user tersebut dikunjungi, apa yang akan terlihat adalah pesan kesalahan halaman tidak ditemukan (404). Hal ini berguna apabila seorang user tidak memerlukan page dan tidak ingin keberadaan user-nya diketahui.

- URL untuk page adalah /page/<user>

11. Referensi: Python handler

Kita telah membahas fitur python handler pada topik **Mengenal SQLiteBoy**, di awal buku ini.

Di bagian ini, kita membahas referensi tentang python handler.

Pastikanlah extended feature telah diaktifkan.

Python handler adalah fitur yang sangat berguna dan memungkinkan SQLiteBoy bekerja sama dengan sistem eksternal/lain.

Apabila python handler tidak disediakan:

- Dalam form, apa yang diinput oleh user akan ditulis dalam database, di mana form dibuat. Apapun operasinya, selalu berefek dalam database tersebut.
- Dalam report, hasil query database, di mana report dibuat, akan menentukan isi report. Apapun querynya, data selalu berasal dari atau berefek dalam database tersebut.

Apabila python handler disediakan, maka kemungkinan menjadi tidak terbatas. Atau setidaknya terbatas pada apa yang disediakan dan didukung oleh python.

Fitur ini diimplementasikan dengan tujuan untuk memudahkan integrasi dengan sistem eksternal atau sistem lain.

Contoh kasus

Berikut adalah beberapa contoh kasus di mana python handler dibutuhkan.

ERP

Perusahaan memutuskan untuk menggunakan satu paket ERP untuk semua operasional. Setidaknya, hampir semua, terutama pada tahap-tahap awal. Konsekuensinya jelas: tidak boleh ada sistem terpisah di luar ERP tersebut.

Permasalahan yang muncul umumnya berhubungan dengan waktu, biaya atau kesiapan pihak yang terlibat, sehingga belum semua modul yang dibutuhkan, tersedia pada ERP tersebut.

Apabila membangun modul-modul tersebut mungkin tidak sederhana, kita bisa selalu memanfaatkan SQLiteBoy, setidaknya sebagai front end. Urusan data entry, setidaknya bisa menggunakan SQLiteBoy. Apa yang diinput oleh user, alih-alih disimpan pada database SQLite lokal, disimpan pada sistem ERP tersebut.

Caranya tentu sangat bergantung pada apa yang disediakan oleh paket ERP tersebut. Apabila API tersedia, maka akan sangat membantu.

Di dalam buku ini, kita akan membahas pula integrasi SQLiteBoy dengan OpenERP.

Sistem database lain

Katakanlah perusahaan telah memilih satu sistem database tertentu, selain SQLite, dan semua data harus disimpan pada database tersebut.

Python mendukung berbagai modul untuk bekerja dengan berbagai sistem database (mungkin tersedia sebagai pustaka pihak ketiga).

Dengan demikian, apa yang diinput oleh user pada form bisa disimpan pada database lain. Report juga bisa menggunakan data yang bersumber dari database lain.

Database SQLite lain

Apa yang diinput oleh user pada form bisa juga disimpan pada database SQLite lain. Begitupun juga dengan report yang bisa menggunakan data yang

bersumber dari database SQLite lainnya.

Apa yang perlu kita lakukan adalah melakukan koneksi ke database SQLite lain tersebut di dalam python handler.

Apapun yang didukung python

Integrasi dengan sistem operasi, hardware ataupun apapun yang didukung oleh python, baik melalui pustaka bawaan ataupun pihak ketiga, dapat dilakukan.

Ketersediaan

Saat ini, python handler hanya tersedia untuk form dan report.

File: sqliteboy_user.py

Semua handler harus disimpan dalam satu file sqliteboy_user.py, di dalam direktori aktif.

Perhatikanlah bahwa satu file sqliteboy_user.py dapat melayani berbagai database, selama berada dalam direktori yang sama dengan database-database tersebut, dan form/report dinamakan berbeda dalam

setiap database.

Form

Setiap form hanya boleh memiliki satu handler. Apabila handler tersebut tersedia, maka akan dipanggil secara otomatis. Tidak ada kode apapun yang perlu ditambahkan dalam definisi form.

Handler untuk form berbentuk satu fungsi python dengan pola nama fungsi berikut:

form_<nama_form>

Apabila handler untuk form tersebut ingin sementara waktu ditiadakan, maka salah satu cara yang bisa dilakukan adalah mengganti nama fungsi tersebut.

Fungsi harus menerima parameter-parameter berikut, dalam urutan:

- user (str): user yang sedang login.
- db (objek database web.py): objek koneksi database aktif. SQL Query dapat dilakukan melalui objek ini.
- parsed (list): data form yang telah diparse.
- user_data (list): apa yang diinput oleh user.

- data (dict): data tambahan yang akan dilewatkan oleh SQLiteBoy sewaktu memanggil fungsi tersebut. Salah satu yang berguna adalah referensi ke UDF SQLiteBoy. Kita akan membahas ini dalam tutorial tersendiri di dalam buku ini.

Fungsi harus mengembalikan satu nilai int. Nilai kembalian ini akan tersedia pada variabel $python_handler, yang bisa digunakan untuk custom message form. Apabila terjadi exception, maka nilai -1 akan diassign ke $python_handler.

Satu hal penting yang perlu diperhatikan adalah: python handler pada form adalah aksi tambahan, dimana fungsi akan dipanggil di akhir form diproses. Python handler pada form tidak akan mengganti handler builtin untuk form.

Integrasi dengan sistem eksternal, sebagai contoh, dilakukan dengan mendapatkan input user dari SQLiteBoy, kemudian menuliskan ke sistem eksternal tersebut.

Report

Setiap report hanya boleh memiliki satu handler. Apabila handler tersebut tersedia, maka akan dipanggil secara otomatis. Tidak ada kode apapun yang perlu ditambahkan dalam definisi report.

Handler untuk report berbentuk satu fungsi python dengan pola nama fungsi berikut:

report_<nama_report>

Apabila handler untuk report tersebut ingin sementara waktu ditiadakan, maka salah satu cara yang bisa dilakukan adalah mengganti nama fungsi tersebut.

Fungsi harus menerima parameter-parameter berikut, dalam urutan:

- user (str): user yang sedang login.
- db (objek database web.py): objek koneksi database aktif. SQL Query dapat dilakukan melalui objek ini.
- parsed (list): data report yang telah diparse.
- user_data (list): apa yang diinput oleh user.
- data (dict): data tambahan yang akan dilewatkan oleh SQLiteBoy sewaktu memanggil fungsi tersebut. Salah satu yang berguna adalah referensi ke UDF SQLiteBoy. Kita akan membahas ini dalam tutorial tersendiri di dalam buku ini.

Fungsi bisa mengembalikan salah satu dari tipe berikut:

- int

- list dari dict (key dari dict akan dianggap sebagai nama kolom)
- objek hasil query database web.py

Satu hal penting yang perlu diperhatikan adalah: python handler pada report adalah pengganti, dimana fungsi akan mengganti SQL Query yang telah didefinisikan. Nilai kembalian dari fungsi akan digunakan sebagai hasil report.

Integrasi dengan sistem eksternal, sebagai contoh, dilakukan dengan membaca dari sistem eksternal tersebut.

12. Referensi: User-defined Profile dan System Profile

Kita telah membahas fitur user-defined profile pada topik **Mengenal SQLiteBoy**, di awal buku ini.

Di bagian ini, kita membahas referensi tentang user-defined profile dan system profile.

Pastikanlah extended feature telah diaktifkan.

SQLiteBoy dirancang untuk menjadi sistem yang general purpose. Oleh karena itu, tidak ada profile yang disediakan spesifik untuk sistem tertentu.

Sebagai contoh, pengaturan profile yang disediakan oleh sistem adalah pengaturan style (tampilan, int), first_name (str), last_name (str), email (str) dan website (str). Ini adalah sesuatu yang umum. Akan tetapi, profile semacam company tidaklah tersedia: bisa saja SQLiteBoy digunakan untuk membangun solusi tertentu yang tidak ada sangkut pautnya dengan pengaturan company.

Lantas, bagaimana kalau kita membutuhkan profile semacam company tersebut? Sebagai contoh, kita membangun suatu sistem perusahaan dimana satu user dapat bekerja pada beberapa perusahaan dalam group, dimana beberapa hal memang terintegrasi dan

beberapa lainnya harus terpisah. Dan, user bisa berpindah dari satu perusahaan ke perusahaan lain dengan cara yang mudah.

Di sinilah user-defined profile berperan. Admin dapat mendefinisikan berbagai profile tambahan yang diinginkan, dan sebagai hasilnya, berbagai field tambahan akan muncul pada halaman profile user, yang nilainya bisa diset oleh user tersebut.

Field tersebut mirip dengan apa yang disediakan oleh form, walaupun jauh lebih terbatas. Sebagai contoh, reference dan default didukung. Dengan demikian, user bisa mengisikan nilai bebas, memilih dari nilai-nilai yang ada (statik ataupun hasil query), dengan nilai default yang bisa diatur.

Apa yang diisikan oleh user dalam user-defined profile dapat dipergunakan dalam form, report, query ataupun lainnya. Ketika profile user disimpan, maka akan langsung berlaku, dan user tersebut tidak perlu logout terlebih dahulu.

User-defined profile merupakan pengaturan sistem dan berlaku untuk semua user. Di halaman profile user (URL: /profile), field-field dalam user-defined profile akan ditandai berbeda.

Untuk mendefinisikan user-defined profile, kita menggunakan kode JSON. Pembahasan di dalam buku ini

akan menggunakan terminologi tipe data seperti halnya pada form.

Pengaturan

- User-defined profile merupakan sebuah list.
- Setiap anggota dalam list tersebut juga harus merupakan list, yang masing-masing memiliki 4 anggota:
  - nama field (str): hanya underscore dan alfanumerik saja
  - label field (str)
  - reference (bacalah referensi form apabila diperlukan)
  - default (bacalah referensi form apabila diperlukan)
- Perhatikanlah bahwa user-defined profile akan selalu disimpan sebagai str. Apabila diperlukan, lakukanlah konversi.

Membaca

Untuk membaca profile yang diisikan oleh user, kita menggunakan fungsi-fungsi yang telah disediakan oleh SQLiteBoy: sqliteboy_x_profile atau sqliteboy_x_my. Fungsi yang terakhir lebih umum digunakan karena berlaku untuk setiap user. Bacalah

referensi UDF apabila diperlukan.

Contoh kode

```
[
  [
     "company",
     "Company",
     "select id as a, name as b from company",
     0
  ],
  [
     "sqliteboy",
     "Happy SQLiteBoy user?",
     [ [0,"no :("], [1,"yes :)"] ],
     1
  ],
  [
     "signature",
     "Signature",
     0,
     ""
  ]
]
```

Contoh penggunaan fungsi

Untuk profile-profile dalam contoh kode sebelumnya, berikut adalah contoh-contoh penggunaan fungsi untuk mendapatkan nilai yang disimpan oleh user.

Mendapatkan profile 'company' untuk user yang sedang login:

```
select sqliteboy_x_my('company')
```

Mendapatkan profile 'company' untuk user 'admin':

```
select sqliteboy_x_profile('admin', 'company')
```

Mendapatkan profile 'company' untuk user 'admin', kemudian mengkonversi nilai yang didapatkan ke int:

```
select
sqliteboy_as_integer(sqliteboy_x_profile('admin',
'company'))
```

Membaca: System Profile

Untuk membaca system profile, kita menggunakan fungsi-fungsi yang disediakan oleh SQLiteBoy:

- sqliteboy_x_profile_system
- sqliteboy_x_my_system

13. Referensi: Script

Kita telah membahas fitur script pada topik **Mengenal SQLiteBoy**, di awal buku ini.

Di bagian ini, kita membahas referensi tentang kode script.

Pastikanlah extended feature telah diaktifkan.

Script memungkinkan pembuatan tabel, form, report dan user-defined profile secara otomatis, berdasarkan apa yang telah didefinisikan dalam sebuah file.

File tersebut kemudian bisa dikopikan ke pengguna SQLiteBoy lain, untuk diupload dan kemudian dijalankan (oleh user dengan hak admin).

Dengan demikian, seseorang bisa mengembangkan solusi berbasis SQLiteBoy dan mengirimkannya ke pihak lain. Baik secara gratis ataupun berbayar.

Untuk mendefinisikan script, kita menggunakan kode JSON. Pembahasan di dalam buku ini akan menggunakan terminologi tipe data seperti halnya pada form.

Pemeriksaan sintaks dan nilai valid

Pada saat script diupload, pemeriksaan sintaks akan dilakukan. Apabila terdapat kesalahan sintaks, script tidak akan disimpan.

Untuk script yang berhasil disimpan: hanya nilai yang valid yang akan dibaca dari script (walaupun isi script akan tetap disimpan apa adanya). Apabila tidak terdapat nilai valid apapun yang terbaca, maka dianggap tidak ada yang didefinisikan dalam script.

Sebagai catatan, nilai valid tidak hanya dari sisi sintaks tapi juga dari sisi apakah nilai tersebut diperlukan atau tidak.

Pemeriksaan sistem

Ketika suatu script ingin dijalankan, SQLiteBoy akan memeriksa dan menentukan apakah script tersebut bisa dijalankan atau tidak.

Script tidak bisa dijalankan apabila tidak ada yang didefinisikan, telah dijalankan sebelumnya atau terdapat satu atau lebih kesalahan ditemukan.

Kesalahan dapat berupa:

- Form yang ingin dibuat telah ditemukan

(pemeriksaan nama). Berhati-hatilah menentukan nama form dalam script.

- Report yang ingin dibuat telah ditemukan (pemeriksaan nama). Berhati-hatilah menentukan nama report dalam script.
- Tabel yang ingin dibuat telah ditemukan dan terdapat kolom yang konflik. Ini akan kita bahas lebih lanjut.

Uninstall

Script yang telah dijalankan tidak dapat di-uninstall.

Script juga dimaksudkan untuk dijalankan hanya sekali.

Kesalahan saat dijalankan

Apabila terjadi kesalahan, untuk alasan apapun, pada saat script sedang dijalankan, maka:

- Untuk setiap tabel yang dibuat, apabila masih kosong, akan dihapus secara eksplisit.
- Kolom yang telah ditambahkan dalam tabel tidak dihapus (karena merupakan pekerjaan yang tidak sederhana di SQLite).
- Transaksi akan di-rollback.

Solusi dalam beberapa script

Walaupun script disimpan di dalam satu file, kita bisa selalu merancang agar suatu solusi dikembangkan dalam beberapa script.

Hal ini tentu saja akan memudahkan:

- Pengembangan solusi lanjutan. Sebagai contoh, dalam script pertama, kita belum menyertakan fitur tertentu dan dapat menyertakannya dalam script kedua, ketiga dan seterusnya.
- Kustomisasi. Sebagai contoh, kita dapat membuat satu sistem yang umum (dipakai oleh sebagian besar calon pengguna), kemudian mengembangkan solusi spesifik untuk pengguna tertentu.

Bagaimana hal ini dapat dilakukan? Kita tahu bahwa solusi berbasis SQLiteBoy dapat melibatkan form, report, user-defined profile dan/atau python handler. Dan, form atau report, secara default akan bekerja dengan data dari tabel-tabel. Dengan demikian:

- Pada saat merancang sebuah sistem yang umum atau terbatas, kita cukup mendefinisikan tabel-tabel yang penting saja untuk sistem tersebut. Kolom-kolom dalam masing-masing tabel juga adalah yang diperlukan saja. Demikian juga dengan form atau report.

- Gunakan nama form, report atau field dalam user-defined profile yang tidak terlalu spesifik.
- Ketika solusi lanjutan atau kustomisasi diperlukan, kita bisa membuat script lain. Di dalam script tersebut, kita bisa menambahkan tabel, menambahkan kolom untuk tabel yang sudah ada atau membuat form/report/user-defined profile dengan nama yang lebih spesifik.

Tentu saja, kita perlu pikirkan agar database tidak menjadi kotor, gara-gara perancangan sistem lanjutan atau kustomisasi yang tidak hati-hati.

Tabel

Berikut adalah beberapa catatan untuk tabel dalam script:

- Lebih dari satu tabel bisa didefinisikan dalam script.
- Untuk setiap tabel, untuk setiap kolom di dalamnya, kita perlu mendefinisikan:
  - nama kolom
  - tipe kolom
  - flag
- Berikut adalah tipe kolom yang valid:
  - integer

    - real
    - char
    - varchar
    - text
    - blob
    - null
- Berikut adalah flag kolom yang valid:
    - 0
    - 1, apabila kolom tersebut merupakan primary key
    - 2, hanya untuk kolom bertipe integer, apabila kolom tersebut bertipe primary key autoincrement.
- Untuk dukungan multiple primary key:
    - Gunakanlah flag 1 untuk setiap kolom yang merupakan primary key.
    - Harap tidak menggunakan flag 1 dan 2 bersamaan dalam satu tabel.
- Saat ini, nilai default tidak didukung. Gunakanlah fitur default pada form untuk membantu.
- Penambahan kolom pada tabel yang ada didukung. Lihatlah pembahasan berikut.

Tabel: yang telah ditemukan

Untuk tabel yang ingin dibuat namun telah ditemukan, pemeriksaan kolom tabel dalam script akan dilakukan. Untuk setiap kolom:

- Apabila kolom tersebut belum ditemukan dalam tabel, maka kolom tersebut, selama valid, tidak akan dianggap kesalahan dan akan ditambahkan dalam tabel.
- Apabila kolom tersebut telah ditemukan dalam tabel, namun memiliki tipe yang sama, maka tidak akan dianggap kesalahan dan akan dilewatkan.
- Apabila kolom tersebut telah ditemukan dalam tabel dan memiliki tipe yang berbeda, maka akan dianggap kolom konflik dan ini merupakan kesalahan (script tidak dapat dijalankan).

Untuk setiap tabel yang ingin dibuat namun telah ditemukan, dan setiap kolom dalam tabel tersebut juga telah ditemukan dan memiliki tipe yang sama (dan oleh karenanya dilewatkan), maka tidak akan ditampilkan.

Form/Report

Berikut adalah beberapa catatan untuk form atau report dalam script:

- Lebih dari satu form atau report bisa didefinisikan dalam script.

- Apabila nama form atau report yang didefinisikan telah ditemukan, maka akan dianggap kesalahan dan script tidak dapat dijalankan.
- Bacalah juga Referensi Form (untuk form) dan Referensi Report (untuk report) apabila diperlukan.

Key

Perhatikanlah bahwa script merupakan sebuah dict. Apa yang kita bahas berikut adalah key pada dict tersebut.

name

Tipe	str
Status	Diperlukan
Contoh	"my script 1"

Merupakan nama script.

tables

Tipe	list dari list

Status	Diperlukan
Contoh	Lihatlah pada contoh kode script

Definisi tabel-tabel yang akan dibuat. Beberapa catatan:

- Harus merupakan list dari list tabel yang akan dibuat, atau berupa list kosong [] apabila script tidak membuat tabel.
- Untuk setiap tabel, berikan nama disertai dengan definisi kolom, sesuai pola:

```
["nama_tabel", [kolom],...]
```

- Untuk setiap [kolom], berikanlah nama, tipe dan flag sebagaimana telah dibahas sebelumnya.

```
["nama", "tipe", flag]
```

forms

Tipe	list dari list
Status	Diperlukan
Contoh	Lihatlah pada contoh kode script

Definisi form-form yang akan dibuat. Beberapa catatan:

- Harus merupakan list dari list form yang akan dibuat, atau berupa list kosong [] apabila script tidak membuat form.
- Untuk setiap form, berikan nama disertai dengan kode form, sesuai pola:

["nama_form", {kode_form}]

- Untuk kode_form, merupakan dict kode form yang valid. Bacalah Referensi Form apabila diperlukan.

reports

Tipe	list dari list
Status	Diperlukan
Contoh	Lihatlah pada contoh kode script

Definisi report-report yang akan dibuat. Beberapa catatan:

- Harus merupakan list dari list report yang akan dibuat, atau berupa list kosong [] apabila script tidak membuat report.
- Untuk setiap report, berikan nama disertai

dengan kode report, sesuai pola:

[“nama_report”, {kode_report}]

- Untuk kode_report, merupakan dict kode report yang valid. Bacalah Referensi Report apabila diperlukan.

profiles

Tipe	list
Status	Opsional
Contoh	Lihatlah pada contoh kode script

Definisi user-defined profile yang akan dibuat. Apabila diberikan, harus merupakan kode user-defined profile yang valid.

info

Tipe	str
Status	Opsional
Contoh	"Script Information"

Merupakan informasi script.

author

Tipe	str
Status	Opsional
Contoh	"(c) Author <email>"

Merupakan informasi developer script. Sertakanlah juga email dan/atau website apabila diperlukan.

license

Tipe	str
Status	Opsional
Contoh	"license"
Contoh lain	“GPL”

Merupakan informasi nama lisensi untuk script.

Contoh kode script 1

```
{
    "name": "my script",
```

```
    "info": "Script Information",
    "author": "(c) Author <email>",
    "license": "GPL",
    "tables": [
                    [
                        "new_table",
                        ["a", "integer", 1],
                        ["b", "integer", 1],
                        ["c", "integer", 1],
                        ["d", "text", 0]
                    ]
                ],
    "forms": [
                ],
    "reports": [
                ]
}
```

Contoh kode script 2

```
{
    "name": "my script 1",
    "info": "Script Information",
    "author": "(c) Author <email>",
```

```
    "license": "GPL",
    "tables": [
                    [
                        "new_table_1",
                        ["a", "integer", 1],
                        ["b", "integer", 1],
                        ["c", "integer", 1],
                        ["d", "text", 0]
                    ],
                    [
                        "new_table_2",
                        ["a", "integer", 2],
                        ["b", "integer", 0],
                        ["c", "integer", 0],
                        ["d", "text", 0]
                    ]
                ],
    "forms": [
                    [
                        "new_form_1",
                        {
                          "title" : "New Form 1",
                          "info"  : "Form Information",
                          "data"  : [
                                      {
                                        "table"     : "new_table_1",
                                        "column"    : "a"
                                      },
                                      {
```

```
                        "table"     : "new_table_1",
                        "column"    : "b"
                      }
                    ],
      "security" : {
                     "run" : ""
                   }
    }
],
[
    "new_form_2",
    {
      "title" : "New Form 2",
      "info"  : "Form Information",
      "data"  : [
                  {
                    "table"     : "new_table_2",
                    "column"    : "c"
                  },
                  {
                    "table"     : "new_table_2",
                    "column"    : "d"
                  }
                ],
      "security" : {
                     "run" : ""
                   }
    }
]
```

```
                ],
    "reports": [
                    [
                        "new_report_1",
                        {
                          "title" : "New Report 1",
                          "info"  : "Report Information",
                          "header": ["a", "b"],
                          "sql"   : "select a,b from new_table_1 a where
a > $input_a or b > $input_b",
                          "data"  : [
                                       {
                                         "key"       : "input_a",
                                         "label"     : "column a >",
                                         "default"   : "0"
                                       },
                                       {
                                         "key"       : "input_b",
                                         "label"     : "column b >",
                                         "default"   : "0"
                                       }
                                     ],
                          "security" : {
                                         "run" : ""
                                       }
                        }
                    ]
                ],
    "profiles": [
                  [
```

```
                    "company",
                    "Company",
                    "select id as a, name as b from company",
                    0
                ],
                [
                    "sqliteboy",
                    "Happy SQLiteBoy user?",
                    [ [0,"no :("], [1,"yes :)"] ],
                    1
                ],
                [
                    "signature",
                    "Signature",
                    0,
                    ""
                ]
            ]
}
```

14. Referensi: Files

Kita telah membahas fitur files pada topik **Mengenal SQLiteBoy**, di awal buku ini.

Di bagian ini, kita membahas referensi files.

Pastikanlah extended feature telah diaktifkan.

Kepemilikan

- Setiap user bisa memiliki sejumlah file, yang didapatkan dari hasil upload (multiple file upload didukung).
- File-file tersebut secara default hanya bisa diakses oleh pemiliknya.

Id dan URL

- Setiap file memiliki id dan SQLiteBoy bekerja berdasarkan id tersebut. Ini berarti, per user, bisa terdapat lebih dari satu file dengan nama yang sama.
- Id sebuah file, ketika file dihapus, secara umum, harusnya tidak dipakai kembali.

- URL untuk mengakses file adalah /fs

Direktori

- Files dalam SQLiteBoy tidak mengenal konsep direktori.

Aksi

- Setiap file yang dimiliki oleh user, apabila didukung, bisa dilangsung ditampilkan dalam web browser. Dalam hal ini, URL adalah /fs?sid=<id>
- Setiap file yang dimiliki oleh user dapat didownload (disposition attachment). Dalam hal ini, URL adalah /fs?download=1&sid=<id>

File sharing

- File dapat di-share agar dapat diakses oleh user lainnya.
- Share dalam hal ini berlaku global. Apabila suatu file di-share, maka semua user dapat mengakses file tersebut.

Error 404

- Apabila suatu file tidak ditemukan (id tidak valid), maka error 404 akan ditampilkan ketika file tersebut diakses.
- Untuk file yang ditemukan, namun tidak di-share, ketika diakses oleh user selain pemilik, juga akan memicu error 404, seolah file tidak ditemukan.

Sebagai header/footer report

- Apabila suatu file gambar digunakan dalam header atau footer report, dan file tersebut ingin tampil dalam report yang dijalankan oleh setiap user, maka file tersebut perlu di-share.

Jumlah file per user

- Jumlah file per user level admin tidak dibatasi.
- Jumlah file per user level standard dapat dibatasi oleh user level admin, secara global, dalam konfigurasi sistem (dibahas dalam bagian tersendiri).
- Nilai default adalah maksimal 10 file per user.

Ukuran file

- Ukuran file milik user level admin tidak dibatasi.
- Ukuran file milik user level standard dapat dibatasi oleh user level admin, secara global, dalam konfigurasi sistem (dibahas dalam bagian tersendiri).
- Nilai default adalah maksimal 1 MB per file.

Penyimpanan

- Setiap file akan disimpan dalam tabel di database, bersama nama dan informasi lainnya.
- Tidak ada file yang disimpan pada file sistem server.
- Backup dapat dilakukan dengan mengopi satu file database.

15. Referensi: Notes

Kita telah membahas fitur notes pada topik **Mengenal SQLiteBoy**, di awal buku ini.

Di bagian ini, kita membahas referensi notes.

Pastikanlah extended feature telah diaktifkan.

- Notes tersedia untuk semua user.
- Jumlah notes per user tidak dibatasi.
- Ketika menyimpan note, user dapat mengisikan judul saja, isi saja, atau keduanya.
- Setiap isi note milik user nantinya dapat digunakan sebagai SQL query (khusus level admin), ataupun sebagai ekspresi calculator.

16. Referensi: User account

Ketika Extended feature diaktifkan, maka SQLiteBoy mengenal konsep user account.

Sebagaimana dibahas sebelumnya, terdapat dua level user:

- admin
- standard

SqliteBoy mendukung lebih dari satu user dengan level admin, di mana nama user pertama untuk level admin tersebut adalah: admin.

SQLiteBoy tidak mengenal konsep group user.

Ganti password

Setiap user yang login dapat mengganti password masing-masing, dengan sebelumnya harus memasukkan password lamanya.

User dengan level admin dapat mengganti password user lainnya, tanpa harus mengetahui password lama user tersebut.

Pengaturan user

Setiap user level admin yang login dapat melakukan pengaturan user dengan mengakses URL: /admin/users, atau melalui link di halaman depan.

17. Referensi: Calculator

Ketika Extended feature diaktifkan, maka SQLiteBoy menyediakan fitur calculator.

Calculator mendukung ekspresi bebas dan berikut adalah batasannya:

- Set karakter yang didukung adalah: 0123456789.-+*/()
- Panjang maksimal ekspresi yang didukung adalah: 36 karakter

Calculator dapat digunakan oleh setiap user.

18. Referensi: Hosts

Apabila extended feature tidak diaktifkan, maka hanya host yang menjalankan SQLiteBoy saja, yang diijinkan untuk akses.

Apabila extended feature diaktifkan, maka host yang diijinkan untuk akses dapat berupa:

- local: host yang menjalankan SQLiteBoy saja. Merupakan nilai default.
- all: semua host
- custom: host-host tertentu saja

URL: /admin/hosts

Untuk pengaturan custom, tuliskanlah semua alamat IP yang diijinkan untuk akses, dipisahkan whitespace.

19. Referensi: System

Pengaturan atau konfigurasi sistem dapat digunakan untuk mengatur beberapa variabel ataupun meng-override beberapa nilai default.

Pastikanlah extended feature telah diaktifkan.

URL: /admin/system

application.title

Section	application
Key	title
Nilai default	
Batasan	Maksimal 32 karakter, HTML tidak diijinkan

Digunakan untuk mengatur judul dari aplikasi, untuk yang tampil pada sisi kiri atas, di template default.

Umumnya, dapat digunakan untuk menampilkan nama perusahaan.

Ini berbeda dengan nilai kembalian fungsi sqliteboy_app_title().

files.max_number

Section	files
Key	maximum number of files per user
Nilai default	10
Batasan	Diisi dengan bilangan

Menentukan jumlah file per user level standard.

files.max_size

Section	files
Key	maximum file size
Nilai default	1048576
Batasan	Diisi dengan bilangan

Menentukan ukuran maksimal file milik user level standard.

scripts.max_size

Section	scripts
Key	maximum script size
Nilai default	32768
Batasan	Diisi dengan bilangan

Menentukan ukuran maksimal script yang bisa diupload.

users.profile

Section	users
Key	user-defined profile
Nilai default	
Batasan	Kode user-defined profile yang valid

Menentukan pengaturan user-defined profile. Akan bekerja sama dengan user-defined profile yang di-set dari script.

messages.all

Section	messages
Key	for all users

Nilai default	
Batasan	HTML tidak diijinkan

Menentukan pesan yang akan ditampilkan untuk semua user, di halaman depan. Untuk pemformatan, dapat menggunakan kode page.

20. Keamanan tambahan

Selain keamanan dari sisi authentikasi, hak user dan aplikasi secara umum, di bagian ini, kita membahas beberapa aspek keamanan tambahan.

Script

Python handler tidak dapat disertakan dalam script, terutama untuk alasan keamanan.

Script dirancang agar bisa dikembangkan oleh pihak ketiga. Dan, oleh karena itu, untuk alasan keamanan, tidak boleh ada kode python yang disertakan (karena dikhawatirkan mengandung kode berbahaya).

Python handler harus dibuat oleh seseorang yang memiliki hak akses ke sistem di mana SQLiteBoy dijalankan, bukan oleh user SQLiteBoy level admin.

Calculator

Sebagaimana dibahas sebelumnya, calculator mendukung ekspresi bebas.

Untuk alasan keamanan, ekspresi calculator dibatasi untuk hanya menerima set karakter tertentu, dengan panjang tertentu, dan perhitungan diimplementasikan dengan SQL Query.

Ekspresi calculator tidak diimplementasikan dengan eval python ataupun yang berpotensi menimbulkan bahaya atau gangguan pada keseluruhan sistem dan/atau user lainnya.

User account

Level admin dapat diberikan kepada user yang membutuhkan diantaranya administrasi database ataupun pengembangan form/report.

Namun, user level admin tidak memiliki hak akses ke sistem yang menjalankan SQLiteBoy.

Kerusakan maksimal yang bisa dilakukan oleh user level admin tersebut (seharusnya) adalah pada file database itu sendiri.

Proteksi tambahan dari sisi sistem yang menjalankan SQLiteBoy, diantaranya adalah dengan menerapkan quota file sistem (apabila diperlukan).

Akses file sistem server

Satu-satunya akses file sistem server yang diijinkan adalah pada direktori dengan nama static (apabila ditemukan), relatif terhadap direktori aktif.

URL yang dipergunakan adalah /static/ (bukan /static).

Dengan demikian, apabila kita menyimpan file di dalam direktori static tersebut, file tersebut akan bisa diakses secara publik.

Bagaimana kalau SQLiteBoy ingin dijalankan, dengan file database yang ditempatkan dalam direktori static, relatif terhadap direktori aktif? SQLiteBoy akan menolak untuk dijalankan.

21. Tutorial: UDF dalam Python handler

Di dalam tutorial ini, kita membahas cara menggunakan berbagai user-defined function yang disediakan SQLiteBoy, di dalam python handler untuk form atau report.

Pastikanlah extended feature telah diaktifkan.

Mari kita lihat kembali fungsi python handler untuk form atau report:

form_<nama_form>(user, db, parsed, user_data, data)

atau

report_<nama_report>(user, db, parsed, user_data, data)

Kita akan menggunakan parameter terakhir: data, yang merupakan sebuah dict.

Untuk mendapatkan referensi ke semua user-defined function yang disediakan oleh SQLiteBoy, akseslah

key udf.

```
udf = data.get('udf')
```

Apa yang kita dapatkan juga berupa dict, dengan key adalah nama fungsi. Contoh penggunaan fungsi sqliteboy_number_to_words():

```
number_to_words =
udf.get('sqliteboy_number_to_words')

print number_to_words('-
12345678912345678912345678 9.123456789', 'en1')
```

22. Tutorial: Membuat partner baru di OpenERP

Di dalam tutorial ini, kita membahas pembuatan partner baru di OpenERP, dengan menggunakan form SQLiteBoy sebagai front end.

Pastikanlah extended feature telah diaktifkan.

Tutorial ini telah diuji coba pada OpenERP 6.1.1 yang berjalan di Microsoft Windows 7 32-bit.

Buat tabel: partner

Pertama-tama, buatlah terlebih dahulu satu tabel dengan nama partner di SQLiteBoy.

Kenapa hal ini diperlukan? Karena kita ingin membuat sebuah form untuk tabel tersebut, di mana kita bisa mengisikan informasi partner yang akan dibuat.

Jumlah kolom	2
Nama tabel	partner

Tipe kolom:

id	integer primary key autoincrement
name	varchar

Buat form: add_partner

Buatlah form dengan nama add_partner, dengan kode form sebagai berikut.

```
{
      "title" : "Add Partner",
      "data"  : [
                  {
                    "table"     : "partner",
                    "column"    : "name",
                    "required"  : 1
                  }
                ],
      "message": ["Error: $result", "Error:
$result", "OK, Partner ID: $python_handler"],
      "security" : {
                     "run" : ""
                   }
}
```

Buat python handler

Buatlah, apabila belum ditemukan sebelumnya, file sqliteboy_user.py di dalam direktori aktif. Kita akan menggunakan python handler untuk berbicara dengan server OpenERP.

Tambahkanlah fungsi berikut ke dalam file tersebut.

```
def form_add_partner(user, db, parsed, form_data,
data):
    import xmlrpclib
    #
    #please edit
    server = 'http://localhost:8069'
    server_common = '%s/xmlrpc/common' %(server)
    server_object = '%s/xmlrpc/object' %(server)
    user = 'admin'
    password = 'admin'
    db = 'test1'
    #
    #
    sock_common =
xmlrpclib.ServerProxy(server_common)
```

```
    uid = sock_common.login(db, user, password)
    #
    sock_object =
xmlrpclib.ServerProxy(server_object)
    fields = form_data[1]
    fields_name = fields.get('name')
    partner = {'name': fields_name}
    partner_id = sock_object.execute(
                db,
                uid,
                password,
                'res.partner',
                'create',
                partner
    )
    #
    return partner_id
```

Catatan

- Sesuaikanlah parameter koneksi ke server OpenERP.
- Kita akan menggunakan XML-RPC untuk berbicara dengan OpenERP. Pastikanlah sistem yang menjalankan server OpenERP telah mengijinkan

akses (contoh: firewall).

- Kita akan menambahkan sebuah res.partner baru, yang hanya memiliki informasi: name. Ini tentu bisa dikembangkan.
- Dengan cara serupa, kita bisa juga membuat objek lainnya.
- Bacalah dokumentasi XML-RPC OpenERP apabila diperlukan.

23. Tutorial: Mencari partner di OpenERP

Di dalam tutorial ini, kita membahas pencarian partner di OpenERP, dimana hasil pencarian akan ditampilkan sebagai report SQLiteBoy.

Pastikanlah extended feature telah diaktifkan.

Tutorial ini telah diuji coba pada OpenERP 6.1.1 yang berjalan di Microsoft Windows 7 32-bit.

Buat report: search_partner

Pertama-tama, buatlah terlebih dahulu satu report dengan nama search_partner, dengan kode berikut.

```
{
  "title" : "Search Partner",
  "header": ["id", "name"],
  "sql"   : "select 1",
  "data"  : [
              {
                "key"       : "name"
```

```
                }
                 ],
  "security" : {
                   "run" : ""
                 }
}
```

Buat python handler

Buatlah, apabila belum ditemukan sebelumnya, file sqliteboy_user.py di dalam direktori aktif. Kita akan menggunakan python handler untuk berbicara dengan server OpenERP.

Tambahkanlah fungsi berikut ke dalam file tersebut.

```
def report_search_partner(user, db, parsed,
form_data, data):
    import xmlrpclib
    #
    #please edit
    server = 'http://localhost:8069'
    server_common = '%s/xmlrpc/common' %(server)
    server_object = '%s/xmlrpc/object' %(server)
```

```
    user = 'admin'
    password = 'admin'
    db = 'test1'
    #
    #
    sock_common =
xmlrpclib.ServerProxy(server_common)
    uid = sock_common.login(db, user, password)
    #
    sock_object =
xmlrpclib.ServerProxy(server_object)
    fields = form_data[0]
    fields_name = fields.get('name')
    #
    search_q = [('name', 'ilike', '%%%s%%' %
(fields_name))]
    search_i = sock_object.execute(
                db,
                uid,
                password,
                'res.partner',
                'search',
                search_q
    )
```

```
    #
    search_f = ['name']
    search_r = sock_object.execute(
                db,
                uid,
                password,
                'res.partner',
                'read',
                search_i,
                search_f
    )
    #
    return search_r
```

Catatan

- Sesuaikanlah parameter koneksi ke server OpenERP.
- Kita akan menggunakan XML-RPC untuk berbicara dengan OpenERP. Pastikanlah sistem yang menjalankan server OpenERP telah mengijinkan akses (contoh: firewall).
- Kita akan mencari res.partner, berdasarkan field: name. Apa yang diisikan oleh user dalam search report SQLiteBoy, kita anggap dapat

berupa bagian dari nama dalam partner. Pencarian dilakukan secara tidak case-sensitive.

- Setelah pencarian dilakukan, kita akan membaca field name untuk setiap res.partner yang ditemukan.
- Setelah itu, apa yang kita dapatkan, kita gunakan sebagai nilai kembalian fungsi.
- Dengan cara serupa, kita bisa juga mencari dan membaca objek lainnya.
- Bacalah dokumentasi XML-RPC OpenERP apabila diperlukan.

24. Tutorial: medical record sederhana

Di dalam tutorial ini, kita membahas pembuatan medical record sederhana, dimana:

- Pasien bisa melakukan pendaftaran.
- Dokter atau tenaga medis terkait bisa mengisikan medical record baru, misal pada saat pemeriksaan pasien.
- Dokter atau tenaga medis terkait bisa melihat keseluruhan medical record milik pasien yang sedang diperiksa (ataupun pasien lainnya).

Pastikanlah extended feature telah diaktifkan.

Semua langkah yang dibahas dalam tutorial ini harus dibuat oleh user level admin. Namun, untuk operasional sehari-hari, gunakanlah user level standard.

Tutorial ini tidak menggunakan python handler.

Catatan: tutorial ini juga tersedia dalam satu script yang siap dijalankan. Script dapat didownload dari http://tedut.com.

Buat tabel: patients

Jumlah kolom	6
Nama tabel	patients

Tipe kolom:

id	integer primary key autoincrement
first_name	varchar
last_name	varchar
date_of_birth	integer
image	blob
note	text

Buat tabel: records

Jumlah kolom	5
Nama tabel	records

Tipe kolom:

id	integer primary key autoincrement
patient_id	integer
date_time	integer
user	varchar
record	text

Buat form: registration

```
{
      "title" : "Patient Registration",
      "data"  : [
                  {
                    "table"     : "patients",
                    "column"    : "first_name",
                    "label"     : "First Name",
                    "required"  : 1
                  },
                  {
                    "table"     : "patients",
                    "column"    : "last_name",
                    "label"     : "Last Name",
                    "required"  : 1
                  },
                  {
                    "table"     : "patients",
                    "column"    : "date_of_birth",
                    "label"     : "Date of Birth
(YYYY-MM-DD)",
                    "required"  : 1
                  },
```

```
                    {
                      "table"     : "patients",
                      "column"    : "image",
                      "label"     : "Photo/Image"
                    },
                    {
                      "table"     : "patients",
                      "column"    : "note",
                      "label"     : "Note"
                    }
                ],
      "security" : {
                     "run" : ""
                   }
}
```

Buat form: medical record

```
{
      "title" : "Add Medical Record",
      "data"  : [
                    {
                      "table"     : "records",
```

```
                    "column"    : "patient_id",
                    "label"     : "Patient",
                    "required"  : 1,
                    "reference" : "select id as a,
id || ' - ' || first_name || ' ' || last_name || '
- ' || date_of_birth as b from patients order by
first_name asc"
                  },
                  {
                    "table"     : "records",
                    "column"    : "record",
                    "label"     : "Record",
                    "required"  : 1
                  },
                  {
                    "table"     : "records",
                    "column"    : "user",
                    "label"     : "User",
                    "required"  : 1,
                    "readonly"  : 1,
                    "default"   :
["sqliteboy_x_user"]
                  },
                  {
                    "table"     : "records",
```

```
                    "column"    : "date_time",
                    "label"     : "Date/Time",
                    "required"  : 1,
                    "readonly"  : 1,
                    "default"   :
["sqliteboy_time3a"]
                  }
                ],
      "security" : {
                     "run" : ""
                   }
}
```

Buat report: patient medical record

```
{
  "title" : "Patient Medical Record",
  "header": ["date_time", "user", "record"],
  "sql"   : "select date_time, user, record from
records where patient_id=$patient_id order by
date_time asc",
  "data"  : [
              {
                "key"       : "patient_id",
```

```
                "label"     : "Patient",
                "reference" : "select id as a, id
|| ' - ' || first_name || ' ' || last_name || ' - '
|| date_of_birth as b from patients order by
first_name asc"
              }
            ],
  "security" : {
                "run" : ""
              }
}
```

Buat user

Adalah merupakan ide yang baik untuk membuat user level standard untuk setiap dokter, perawat, resepsionis, kasir ataupun lainnya.

Catatan

- Form registration digunakan untuk pendaftaran pasien.
- Form medical_record digunakan untuk menambahkan medical record baru.
- Report patient_medical_record digunakan untuk melihat medical record pasien.

- Untuk informasi umur pasien seperti sekian tahun sekian bulan sekian hari, kita bisa menggunakan user-defined function sqliteboy_time6().

25. Tips

Hyperlink dan Javascript pada label

Label pada form/report mendukung HTML. Kita akan menggunakan hyperlink dan javascript pada label untuk menjalankan form/report.

Sebagai contoh, pada form1, kita memiliki satu kolom a pada tabel1. Label saat ini adalah: column a.

```
{
  "table"     : "table1",
  "column"    : "a",
  "label"     : "column a",
  "required"  : 1,
  "reference" : [ ["0", "NO"], ["1", "YES"] ],
  "default"   : "1"
}
```

Mari kita asumsikan bahwa kita ingin menggunakan hyperlink sebagai label kolom. Dan, pada saat hyperlink tersebut diklik, report1 akan dijalankan, dan nilai kolom a akan dilewatkan ke report1 (untuk key input_a_a).

Kita dapat memodifikasi label:

```
{
  "table"     : "table1",
  "column"    : "a",
  "label"     : "<a href='#'
onclick=\"window.location.href='/report/run/report1
?input_a_a='+document.getElementsByName('a')
[0].value\">column a</a>",
  "required"  : 1,
  "reference" : [ ["0", "NO"], ["1", "YES"] ],
  "default"   : "1"
}
```

Catatan:

- URI untuk menjalankan report: /report/run/<nama_report>
- URI untuk menjalankan form: /form/run/<nama_form>
- input_a_a didefinisikan pada report1 sebagai berikut:

```
{
  "key"       : "input_a_a",
  "label"     : "column a equals",
  "reference" : [ ["0", "NO"], ["1", "YES"] ],
  "default"   : "1"
}
```

26. Materi training: python dasar

Materi training singkat, untuk dasar-dasar python, mengacu ke python 2.x, versi 2.5.

Penulisan source code

a) Indentasi adalah 4 karakter.

b) Untuk indentasi, gunakan hanya spasi, jangan gunakan TAB.

c) Panjang baris maksimal adalah 79 karakter. Pindah baris langsung dalam kurung dan hanya gunakan backslash (\) apabila diperlukan.

d) Gunakan dua baris kosong antara class.

e) Gunakan satu baris kosong antara fungsi.

f) Import setiap modul dilakukan per baris, dengan urutan: standard library, pustaka pihak ketiga, pustaka lokal.

g) Struktur program Python (opsional):

   1. Shebang Line
   2. Komentar dan docstring
   3. Import
   4. Global dan konstanta
   5. Class dan fungsi

h) Aturan nama:

1. Paket: lowercase
2. Modul: lowercase, underscore bisa digunakan kalau diperlukan.
3. Class: kapitalisasi per kata (huruf pertama setiap kata menggunakan kapital)
4. Exception: kapitalisasi per kata (huruf pertama setiap kata menggunakan kapital). Gunakan tambahan Error untuk menandakan kesalahan.
5. Fungsi dan method class: lowercase, setiap kata dipisahkan underscore.
6. Variabel: lowercase, setiap kata dipisahkan underscore.
7. Konstanta: lowercase atau UPPERCASE, setiap kata dipisahkan underscore.

i) Case sensitive, huruf kecil dan besar dibedakan.

Sekilas tentang Python

a) Open Source

1. Sesuai dengan sertifikasi Open Source Initiative.
2. Kompatibel dengan GPL, menurut Free Software Foundation. Walau demikian, tidak ada pembatasan copyleft GPL.
3. Bebas digunakan, termasuk untuk produk

proprietary.

4. Selengkapnya, http://www.python.org/psf/license/.

b) Sintaks sederhana, jelas, fleksibel, mudah dipelajari.

c) Mendukung multi-paradigma, salah satunya Object-oriented.

d) Tipe data very high level.

e) Standard library yang sangat kaya.

f) Berjalan di sangat banyak sistem.

g) Dapat diextend dengan C/C++

Interpreter Python (interaktif)

a) Path default executable python:

1. Windows: c:\pythonXY\python.exe. XY adalah versi Python. Contoh: c:\python25\python.exe
2. Linux: /usr/bin/python atau /usr/bin/pythonX.Y (umumnya menggunakan symlink /usr/bin/python). XY adalah versi Python.

b) Untuk menjalankan interpreter Python:

1. Windows: Masuk ke command prompt, masuk ke direktori instalasi Python, jalankan python.exe. Dapat pula mengakses dari Start Menu. Daftarkanlah direktori instalasi Python ke PATH apabila diperlukan.

2. Linux: Masuk ke terminal, berikan perintah: 'python' (tanpa tanda kutip).

c) Keluar dari sesi interaktif interpreter Python:

1. Windows: Ctrl-Z, ENTER.

2. Linux: Ctrl-D

d) Untuk mencetak ke standard output, gunakan print. Contoh akan dibahas dalam sesi training.

e) Untuk membaca string dari standard input, raw_input() bisa digunakan. Contoh akan dibahas dalam sesi training.

f) Beberapa fungsi lain:

1. abs()

2. chr()

3. dir()

4. max()

5. min()

6. pow()

Catatan untuk instruktur:

- Di dalam sesi interactive, instruktur akan menjelaskan berbagai contoh penggunaan, termasuk menggunakan python sebagai kalkulator.
- Demo tipe data dan modul bisa dilakukan.
- Demo fungsi lain bisa dilakukan, apabila memungkinkan.

Script Python

a) Simpan dengan ekstensi nama file py. Contoh: hello.py.

b) Shebang

   1. #, !, diikuti Path ke executable Python
   2. Contoh di Linux: #!/usr/bin/python
   3. Contoh di Windows: #!c:\python25\python.exe
   4. Di Linux, bisa gunakan bantuan env, sehingga shebang line menjadi #!/usr/bin/env python
      a) Sesuaikan path ke env

c) Jalankan di command line: python <script.py>

d) Di Linux, berikan hak akses executable dengan perintah:

   1. chmod +x <script.py>
   2. Selanjutnya, bisa dijalankan langsung dengan ./script.py atau script.py (apabila direktori aktif terdaftar di $PATH) dari prompt.

Tipe Builtin dan operator

Tipe	Detil	Catatan
None	None Object	

Number	• Boolean (bool)
	  • True atau False
	• Integer (int)
	  • Batas int bisa dilihat pada sys.maxint
	• Long integer (long)
	  • Gunakan suffix L atau l
	  • Batas long sesuai memori
	• Floating point (float)
	  • pecahan
	  • bisa ditulis dalam e atau E
	• Complex (complex)
	  • bilangan kompleks
	  • memiliki atribut real (real) dan imag (imajiner).	

	• imajiner bisa dituliskan dengan suffix j atau J. • dir: conjugate, imag, real	
Set	• Set (set): mutable • dir: add, clear, copy, difference, difference_update, discard, intersection, intersection_update, issubset, issuperset, pop, remove, symmetric_difference, symmetric_difference_update, union, update • Frozen Set (frozenset): immutable • dir: copy, difference, intersection, issubset, issuperset, symmetric_difference, union	• Unordered • Semua item dalam set harus immutable • Dapat dibuat dari item iterable • Unik • Dapat dioperasikan sebagai himpunan
Sequence	• String (str)	Memiliki sifat

	• immutable
	• dapat dibentuk dengan pasangan:
	  • kutip tunggal (')
	  • kutip ganda ('')
	  • triple kutip tunggal (' ' ')
	  • triple kutip ganda ('' '' '')
	• Pasangan triple-kutip bisa mengembed newline.
	• Berlaku penggunaan escape character.
	• Dapat dituliskan dalam bentuk raw string (diawali r atau R). Contoh: r'c:\window'
	• Raw string, apabila diakhiri dengan backslash (\), harus berjumlah genap.
	• dir: capitalize, center, count, decode, encode, endswith, expandtabs, find, index, isalnum, isalpha, isdigit, islower, isspace,
		iterable

	istitle, isupper, join, ljust, lower, lstrip, partition, replace, rfind, rindex, rjust, rpartition, rsplit, rstrip, split, splitlines, startswith, strip, swapcase, title, translate, upper, zfill. • Unicode String (unicode) • immutable • Diawali dengan u atau U, diikuti 4 digit hexa kode karakter unicode. • Berlaku penggunaan escape character. • Bisa diisi dengan \N{Name}, dimana Name sesuai aturan unicode. Contoh: \N{Copyright Sign}. • Apabila menggunakan raw string, ditulis dengan ur. • dir: method pada str, ditambah isdecimal dan isnumeric.	

	• Tuple (tuple)
	  • immutable
	  • Akses lebih cepat
	  • Dapat dibuat dengan tuple(), atau dengan menempatkan anggota dalam kurung (dan).
	  • Koma tambahan bisa dituliskan setelah anggota terakhir. Contoh: (1,2,3,)
	  • Tuple dengan satu anggota harus dituliskan dengan menambahkan koma. Contoh: (1,) dan bukan (1)
	• List (list)
	  • mutable
	  • Lihat range()
	  • Dapat dibuat dengan list(), atau dengan menempatkan anggota dalam kurung [dan].
	  • Koma tambahan bisa dituliskan setelah anggota terakhir. Contoh: [1,2,3,]	

	• dir: append, count, extend, index, insert, pop, remove, reverse, sort. • Xrange (xrange) • immutable • Dibuat dengan xrange() • Untuk perulangan, relatif lebih cepat dari penggunaan range().	
Mapping	• Dictionary (dict) • Key harus berupa tipe immutable • Value dapat berupa hampir semua object Python • Dapat dibuat dengan dict(), atau dengan menempatkan key:value dalam kurung { dan }. • dir: clear, copy, fromkeys, get, has_key, items, iteritems, iterkeys, itervalues, keys, pop, popitem, setdefault, update, values	Unordered

File	• File (file)
	  • Dibuka dengan open(name[, mode[, buffering]]) atau file(name[, mode[, buffering]])
	  • Mode file: r(read, default), w(write), a(append)
	  • Ditutup dengan method close() objek file
	  • dir: close, closed, encoding, fileno, flush, isatty, mode, name, newlines, next, read, readinto, readline, readlines, seek, softspace, tell, truncate, write, writelines, xreadlines	

Operator

- Perbandingan boolean
  - <, lebih kecil
  - <=, lebih kecil sama dengan

    - >, lebih besar
    - >=, lebih besar sama dengan
    - ==, sama dengan
    - !=, tidak sama dengan
- Operator Logical
    - not, logical negasi
    - and, logical and
    - or, logical or
- Operator aritmatika
    - *, perkalian
    - /, pembagian
    - //, pembagian integer
    - %, sisa bagi
    - +, penjumlahan
    - -, pengurangan
- Operator bit
    - ~, bitwise complement
    - <<, shift left
    - >>, shift right
    - &, bitwise and
    - |, bitwise or

    - ^, bitwise xor
- Operator pada sequence
    - in: terdapat di dalam. Gunakan not in untuk kebalikannya.
- Operator tambahan string:
    - %: pemformatan

Index dan Slice

Beberapa sequence seperti string, unicode string, list dan tuple mendukung index dan slicing, dengan sintaks:

- s[i]: item ke i dari sequence s, dimulai dari 0
- s[i:j]: item ke i sampai j dari sequence s
- s[i:j:k]: item ke i sampai j dari sequence s, dengan step k

Catatan untuk i,j dan k:

- Apabila nilai i dan j negatif, maka index relatif dari akhir sequence.
- Jika i tidak diberikan, maka default ke 0
- Jika j tidak diberikan, maka default ke len(s)
- Jika k tidak diberikan, maka default ke 1. Nilai k tidak bisa diisi dengan 0.

- Jika i lebih besar atau sama dengan j, maka menghasilkan slice kosong.

Untuk tipe xrange, index didukung namun slicing tidak didukung.

Lain-lain

- Beberapa fungsi:
  - type(object): mengembalikan tipe object
  - id(object): mengembalikan alamat memori object
  - ininstance(object, class-or-type-or-tuple): mengembalikan apakah object merupakan instance dari class.
- Kategori tipe juga mencakup Callable (dapat dipanggil), Modules (setelah diimport), Classes dan Type.

Seleksi/Kondisi

Sintaks if

if <expression>:

```
    <statement>
    <statement>
    ..
[elif <expression>:
    <statement>
    <statement>
    ..
]
[else:
    <statement>
    <statement>
    ..
]
```

Contoh if:

```
>>> a=10
>>> if a > 5:
...     print 'a besar dari 5'
...
a besar dari 5
```

Catatan:

- pass dapat digunakan untuk blok kosong.

- Beberapa contoh akan diberikan pada sesi training.
- Python tidak mendukung switch/case

Perulangan

Sintaks for

```
for <target_list> in <expression_list>:
    <statement>
    <statement>
    ...
[else:
    <statement>
    <statement>
    ..
]
```

Contoh for:

```
>>> for i in range(1,10,2):
...     print i
...
1
```

```
3
5
7
9
```

Sintaks while

```
while <expression>:
    <statement>
    <statement>
    ...
[else:
    <statement>
    <statement>
    ..
]
```

Contoh while:

```
>>> a=1
>>> while a<5:
...     print a
...     a += 1
...
```

```
1
2
3
4
```

Catatan:

- pass dapat digunakan untuk blok kosong.
- Beberapa contoh akan diberikan pada sesi training.

Fungsi

Deklarasi fungsi

```
def <func_name>([parameter_list]):
    <statement>
    <statement>
    ...
    [return <return_value>]
```

Catatan:

- ketika return tidak didefinisikan, None akan dikembalikan

Documentation String (docstring)

Setiap fungsi dapat memiliki docstring (documentation string), yang umumnya mendeskripsikan bagaimana cara menggunakan fungsi tersebut. Docstring dapat dituliskan sebagai string langsung setelah deklarasi fungsi. Contoh:

```
def test():
...     'docstring fungsi test'
...     print 'test'
...
```

Untuk mengetahui docstring fungsi test, akseslah atribut __doc__ dari fungsi test. Contoh:

```
>>> test.__doc__
'docstring fungsi test'
```

Docstring yang didefinisikan juga berguna dalam penggunaan bersama fungsi help() di sesi interactive:

```
>>> help(test)
Help on function test in module __main__:
```

```
test()
    docstring fungsi test
```

Pemanggilan fungsi

Pemanggilan fungsi harus melihatkan penggunaan (), sesuai argumen fungsi. Tanpa (), python tidak akan menganggap sintaks tersebut sebagai sintaks yang salah, tetapi tidak akan dikerjakan.

Contoh yang salah:

```
>>> test
<function test at 0xb7c305dc>
```

Contoh yang benar:

```
>>> test()
test
```

Variabel global

Setiap fungsi akan memiliki variabel lokal sendiri. Namun, ada kalanya, akses ke variabel global diperlukan.

Untuk mengassign nilai tertentu ke variabel global, gunakan keyword global.

Contoh berikut dimaksudkan untuk mengubah variabel global x menjadi 20, namun tidak bekerja:

```
>>> x=10
>>> def testx():
...     x=20
...     print x
...
>>> x
10
>>> testx()
20
>>> x
10
>>>
```

Contoh yang benar adalah:

```
>>> x=10
>>> def testx():
...     global x
...     x=20
```

```
...     print x
...
>>> x
10
>>> testx()
20
>>> x
20
>>>
```

Walau demikian, untuk sekedar mereferensi ke variabel global, keyword global tidak diperlukan.

Argumen fungsi

Argumen fungsi bisa diberikan, tanpa harus menyebutkan tipe data. Apabila lebih dari 1 argumen, maka deretkanlah dengan dipisahkan oleh koma. Pemanggilan fungsi selanjutnya harus disesuaikan dengan argumen yang didefinisikan, kecuali default argument digunakan (lihat pembahasan berikutnya).

Contoh 1:

```
>>> def kuadrat(x):
...     return x*x
```

```
...
>>> hasil=kuadrat(12)
>>> print hasil
144
```

Contoh 2:

```
>>> def kali(a,b):
...     return a*b
...
>>> hasil=kali(2,4)
>>> print hasil
8
```

Argumen juga bisa dipanggil dengan menyebutkan nama (keyword) pada parameter formal, sehingga dapat dipanggil dengan urutan yang tidak sesuai dengan saat deklarasi. Contoh:

```
>>> def kali2(a,b):
...     print '(%d x %d) = %d' %(a, b, a*b)
...
>>> kali2(b=20, a=10)
(10 x 20) = 200
>>>
```

Ketika menggunakan keyword argument, pastikan kesalahan-kesalahan seperti berikut tidak dilakukan:

- non-keyword argument diberikan keyword argument
- duplikasi argumen
- salah memberikan keyword

Argumen default

Argumen default memungkinkan pengguna fungsi untuk memanggil fungsi dengan argumen yang lebih sedikit/sederhana dalam kondisi normal, namun memiliki keleluasaan untuk memanggil fungsi dengan semua argumen diberikan.

Argumen default digunakan apabila sebuah argumen memiliki nilai tertentu yang umum (digunakan oleh pemanggil fungsi), namun tetap dimungkinkan untuk diberikan nilai lain.

Argumen default sangat umum digunakan di Python.

Sebagai contoh, kita mendefinisikan sebuah fungsi cetak_nama, dengan dua argumen:

- name: nama yang akan dicetak

- prefix: prefix pencetakan

Dalam kondisi normal, fungsi akan selalu mencetak tulisan 'Nama Anda adalah ' (prefix) diikuti oleh name yang diberikan. Apabila prefix ingin diganti, pengguna fungsi tetap dapat melakukannya.

```
>>> def cetak_nama(name, prefix='Nama Anda adalah
'):
...     print prefix + name
...
>>> cetak_nama('piton')
Nama Anda adalah piton
>>> cetak_nama('piton','Nama=')
Nama=piton
```

Argumen *arg (tuple)

Ketika parameter formal dituliskan dalam bentuk *arg, maka fungsi akan menerima argumen berupa tuple. Contoh:

```
>>> def testt(*t):
...     for i in t:
...         print i
```

```
...
>>> testt(1,2,3,4,5)
1
2
3
4
5
>>> testt('halo','apa','kabar')
halo
apa
kabar
```

Dengan cara seperti ini, jumlah argumen tidak didefinisikan secara kaku.

Argumen **arg (dictionary)

Ketika parameter formal dituliskan dalam bentuk **arg, maka fungsi akan menerima argumen berupa dictionary. Contoh:

```
>>> def testd(**arg):
...     keys = arg.keys()
...     for k in keys:
...         print '%s=>%s' %(k, arg[k])
```

```
...

>>> testd(nama='piton', umur=28, kabar='Baik')
nama=>piton
kabar=>Baik
umur=>28
>>>
```

Dengan cara seperti ini, argumen tidak didefinisikan secara kaku.

Class

Berikut adalah sintaks class:

```
class <class_name> ([baseclasses]):
    <statement>
    <statement>
    ...
```

Catatan:

- Dideklarasikan dengan keyword class

- Class mendukung docstring seperti halnya fungsi
- setiap definisi method class akan menggunakan keyword self sebagai argumen pertama method, yang dapat digunakan untuk merujuk ke diri sendiri.
- Untuk merujuk ke anggota class, gunakanlah kata kunci self.
- Constructor class adalah method dengan nama __init__.
- Python mendukung multiple inheritance
- Python menggunakan name mangling untuk identifier yang diawali oleh paling tidak dua underscore dan diakhiri paling banyak satu underscore. Name mangling tersebut dapat digunakan sebagai 'private variable'.
- Mengenal dua tipe class: old-style (classic) dan new-style (diperkenalkan sejak Python versi 2.2). Default adalah old-style pada python 2.x. Untuk membuat class new-style, gunakan base class berupa class new-style atau object (apabila base class tidak diperlukan).

Contoh 1, class kosong:

```
>>> class MyClass1:
...     pass
...
>>> c1 = MyClass1()
```

```
>>> type(c1)
<type 'instance'>
>>> c1
<__main__.MyClass1 instance at 0xb7bf1fcc>
```

Contoh 2, class dengan docstring dan constructor:

```
>>> class MyClass2:
...     'keterangan MyClass2'
...     def __init__(self, x):
...         print 'Inisialisasi...'
...         print x
...
>>> c2 = MyClass2(10)
Inisialisasi...
10
>>> c2.__doc__
'keterangan MyClass2'
>>>
```

Contoh 3, atribut class:

```
>>> class MyClass3:
...     def __init__(self, x):
```

```
...         self.x = x
...     def method1(self):
...         print '-' * self.x
...     def method2(self):
...         self.method1()
...
>>> c3 = MyClass3(5)
>>> c3.x
5
>>> c3.method1()
-----
>>> c3.x = 10
>>> c3.method1()
----------
>>> c3.method2()
----------
>>>
```

Contoh 4, inheritance sederhana:

```
>>> class Base:
...     def method1(self):
...         print 'base....'
```

```
...
>>> class Derived1(Base):
...     def method2(self):
...         print 'derived1...'
...
>>> class Derived2(Base):
...     def method1(self):
...         print 'method1 of derived2...'
...     def method2(self):
...         print 'derived2...'
...
>>> c4 = Derived1()
>>> c4.method1()
base....
>>> c4.method2()
derived1...
>>>
>>> c5 = Derived2()
>>> c5.method1()
method1 of derived2...
>>> c5.method2()
derived2...
>>>
```

Contoh 5, name mangling sederhana:

```
>>> class Mangling:
...     def __init__(self, x):
...         self.__x = x
...     def method1(self):
...         print self.__x
...
>>> c6 = Mangling(10)
>>> c6.__x
Traceback (most recent call last):
  File "<stdin>", line 1, in <module>
AttributeError: Mangling instance has no attribute
'__x'
>>> c6.method1()
10
>>> c6.__x=20
>>> c6.method1()
10
>>>
```

Contoh 6, class sebagai struct/record:

```
>>> class Struct1:
...     pass
```

```
...
>>> c7 = Struct1()
>>> c7.name = 'test'
>>> c7.address = 'test address'
>>>
>>> print c7.name
test
>>> print c7.address
test address
```

Modul-modul

Python datang dengan sangat baik modul siap pakai. Pengguna juga dapat menginstall modul tambahan dari pihak ketiga, ataupun menggunakan modul yang dibuat sendiri. Modul-modul bisa pula dikelompokkan menjadi package.

Modul dapat digunakan setelah diimport. Contoh penggunaan modul os untuk mendapatkan nama os:

```
>>> import os
>>> os.name
'posix'
```

Nama yang lebih pendek bisa digunakan, dengan import <module> as <alias>, sebagai contoh:

```
>>> import platform as p
>>> p.uname()
('Linux', 'nop1', '2.6.21.5-smp', '#2 SMP Tue Jun 19 14:58:11 CDT 2007', 'i686', 'Intel(R) Celeron(R) CPU 2.50GHz')
>>>
```

Statement import selengkapnya:

- import <module> [as <name>] (, <module> [as <name>])
- from <relative_module> import <identifier> [as <name>] (, <identifier> [as <name>])
- from <relative_module> import (<identifier> [as <name>] (, <identifier> [as <name>])... [,])
- from <module> import *

Contoh modul

Beberapa contoh modul:

time

bekerja dengan waktu.

Epoch = waktu dimulai (1 Januari 1970, pukul 00:00)

- Dapatkan detik sejak epoch:

```
time.time()
1228025944.210458
```

- Informasi detil waktu sejak epoch (GMT):

```
time.gmtime(-100000000)
(1966, 10, 31, 14, 13, 20, 0, 304, 0)

time.gmtime(0)
(1970, 1, 1, 0, 0, 0, 3, 1, 0)

time.gmtime()
(2008, 11, 30, 6, 19, 48, 6, 335, 0)
```

- Informasi detil waktu sejak epoch (lokal):

```
time.localtime()
(2008, 11, 30, 15, 43, 36, 6, 335, 0)
```

- Parse time dari string (format bisa baca referensi modul time)

```
time.strptime('2008-11-12', '%Y-%m-%d')
(2008, 11, 12, 0, 0, 0, 2, 317, -1)
```

- Mengembalikan string time dari detik sejak epoch (lokal):

```
time.ctime(1000)
'Thu Jan  1 07:16:40 1970'
```

- Mengembalikan string time dari sequence time (lokal):

```
time.asctime((2008,10,11,23,11,22,0,0,0))
'Mon Oct 11 23:11:22 2008'
```

- Mengembalikan detik sejak epoch dari sequence time (lokal):

```
time.mktime((2008,10,11,23,11,22,0,0,0))
1223741482.0
```

- Memformat waktu dari sequence time (GMT atau lokal, bacalah referensi modul time):

```
time.strftime('%d-%m-%Y',
time.localtime(1000000000))
'09-09-2001'
```

- Dapatkan nama timezone (gunakan index 0, apabila non-DST):

```
time.tzname
('WIT', 'WIT')
```

- Dapatkan selisih detik dari GMT:

```
time.timezone
-25200
```

- Menunda eksekusi selama beberapa waktu tertentu (dalam detik):

```
time.sleep(1)
time.sleep(0.5)
```

random

bekerja dengan bilangan acak.

- Mendapatkan bilangan acak antara 0.0 dan 1.0:

```
random.random()
0.47981142386967368
```

- Mendapatkan bilangan acak antara dua integer (termasuk):

```
random.randint(1000,9999)
9923
```

- Memilih bilangan acak dari range():

```
random.randrange(1, 100, 2)
51
```

- Memilih bilangan acak dari sequence:

```
random.choice([1,2,3,4])
2
```

- Mengacak sequence:

```
x=[1,2,3,4,5,6]
random.shuffle(x)
x
[1, 4, 2, 5, 3, 6]
```

sys

akses ke objek yang digunakan atau berhubungan dengan interpreter.

contoh: argv, maxint, version, platform, prefix, executable

os

rutin sistem operasi

contoh: path, name

platform

identitas platform

contoh: uname, system

datetime

bekerja dengan tipe datetime

calendar

fungsi pencetakan kalender

math

fungsi-fungsi matematika

base64

encoding Base16, Base32, Base64

sha

SHA-1 message digest

binascii

konversi binary-ascii

csv

baca tulis file csv

glob

pathname pattern expansion

linecache

akses ke baris tertentu dalam file

md5

MD5 message digest

tarfile

bekerja dengan file tar, tar.gz dan tar.bz2

Selengkapnya, lihatlah index modul pada dokumentasi Python.

Search path

Ketika suatu modul diimport, Python akan mencari ke:

- direktori aktif
- daftar direktori pada variabel PYTHONPATH

- default path, lokasi instalasi python

Compiled module

Ketika suatu modul berhasil diimport, versi byte-compiled modul (file bernama sama dengan modul, namun dengan ekstensi .pyc) akan coba dibuat. File byte-compiled tersebut dapat diload lebih cepat.

Nama module

Setiap modul python memiliki nama masing-masing, yang dapat diakses dari properti __name__. Contoh:

```
>>> import os
>>> os.__name__
'os'
>>> import random
>>> random.__name__
'random'
```

Sebuah nama spesial __main__ akan didefinisikan apabila modul dijalankan standalone. Dengan memeriksa apakah __name__ == '__main__', kita bisa memeriksa apakah modul diimport atau dijalankan standalone.

Contoh:

```
$ cat module1.py
#!/usr/bin/env python

if __name__ == '__main__':
    print 'Dijalankan standalone'
else:
    print 'Diimport'

$ python module1.py
Dijalankan standalone

>>> import module1
Diimport
```

Exception

Kerjakan apa yang ingin dikerjakan, dan apabila terjadi kesalahan, kita siapkan handlernya.

Try...except...finally

Sintaks try...except...finally:

```
try:
    <statement>
    <statement>
    ...
except [expression [, target]]:
    <statement>
    <statement>
    ...
[else:
    <statement>
    <statement>
]
[finally:
    <statement>
    <statement>
```

]

Alur kerja:

- Statement diantara try dan except akan dikerjakan
- Apabila tidak terdapat kesalahan, maka klausa except akan dilewati
- Apabila terjadi kesalahan, sisa perintah yang masih ada di dalam try akan dilewati dan apabila exception yang bersesuaian ditemukan, maka klausa except tersebut akan dikerjakan.
- Apabila terjadi kesalahan namun tidak dihandle, maka secara otomatis, akan dilewatkan ke blok try yang lebih luar, dan apabila tidak dihandle juga, maka kesalahan tersebut merupakan unhandled exception dan eksekusi akan berhenti sesuai dengan kesalahannya.

Catatan:

- Klausa else akan dikerjakan, apabila diberikan, dan tidak ada kesalahan yang terjadi. Berguna untuk perintah yang akan dikerjakan apabila exception tidak terjadi.
- try...except...finally berlaku mulai Python v2.5. Sebelumnya, try..except harus ditempatkan bersarang di try...finally
- Blok finally akan selalu dikerjakan

Contoh tanpa exception:

```
>>> 1/0
Traceback (most recent call last):
  File "<stdin>", line 1, in <module>
ZeroDivisionError: integer division or modulo by zero
```

Contoh dengan exception sederhana:

```
>>> try:
...     1/0
... except ZeroDivisionError:
...     print 'Kesalahan: pembagian dengan nol'
...
Kesalahan: pembagian dengan nol
```

Contoh dengan exception, melibatkan else:

```
>>> try:
...     1/1
... except ZeroDivisionError:
...     print 'Kesalahan: pembagian dengan nol'
... else:
...     print 'Tidak ada kesalahan yang terjadi'
...
1
```

```
Tidak ada kesalahan yang terjadi
```

Contoh dengan exception, else dan finally:

```
>>> try:
...     1/1
... except ZeroDivisionError:
...     print 'Kesalahan: pembagian dengan nol'
... else:
...     print 'Tidak ada kesalahan yang terjadi'
... finally:
...     print 'Pembagian selesai'
...
1
Tidak ada kesalahan yang terjadi
Pembagian selesai
```

Try...finally

Sintaks try...finally

```
try:
    <statement>
    <statement>
```

```
    ...
finally:
    <statement>
    <statement>
    ...
```

Catatan:

- try...finally dapat berguna sebagai cleanup action, apapun kondisi yang terjadi.

Informasi exception

Apabila exception terjadi, informasi tertentu mungkin akan tersedia, sesuai dengan jenis exceptionnya. Untuk mendapatkan, lewatkan argumen pada exception, seperti contoh berikut.

```
>>> try:
...     1/0
... except ZeroDivisionError, e:
...     print 'Kesalahan: ',  e
...
Kesalahan: integer division or modulo by zero
```

Kita dapat pula mengakses atribut message apabila

dibutuhkan. Contoh:

```
>>> try:
...     1/0
... except ZeroDivisionError, e:
...     print 'Kesalahan: ' + e.message
...
Kesalahan: integer division or modulo by zero
```

Membangkitkan exception

Untuk membangkitkan kesalahan tertentu, gunakanlah raise. Argumen opsional bisa diberikan.

```
>>> raise NameError
Traceback (most recent call last):
  File "<stdin>", line 1, in <module>
NameError

>>> raise NameError, 'Ini detil kesalahan'
Traceback (most recent call last):
  File "<stdin>", line 1, in <module>
NameError: Ini detil kesalahan
```

File

Operasi file umumnya melibatkan pembukaan file, melakukan operasi tertentu, dan kemudian menutup file terbuka tersebut.

Membuka dan menutup file

Untuk membuka file, gunakanlah fungsi:

```
open(path [,mode [,buffersize]])
```

Catatan:

- path adalah path file yang ingin dibuka
- mode dapat bernilai:
  - r: membuka file untuk dibaca
  - w: membuka file untuk ditulis. Apabila file yang ingin dibuka telah terdapat di filesystem, maka isinya akan dihapus. Apabila file tidak ditemukan, maka akan dibuat secara otomatis.
  - a: membuka file untuk diupdate. Isi file yang sudah ada tidak dihapus.
  - r+: membuka file untuk dibaca dan ditulis. Isi file yang sudah ada tidak dihapus.
  - W+: membuka file untuk ditulis dan dibaca. Isi file yang sudah ada akan dihapus.

  - a+: membuka file untuk dibaca dan ditulis. Isi file yang sudah ada tidak dihapus.
  - b: apabila ditambahkan ke salah satu dari r, w, atau a, maka akan membuka file dalam modus binary.
  - U: apabila ditambahkan ke salah satu dari r, w atau a, maka akan mengaplikasikan universal newline. Newline pada setiap platform bisa berbeda-beda. Contoh: Linux menggunakan \n, Windows menggunakan \r\n.
- buffersize: ukuran buffer

Fungsi open yang berhasil akan mengembalikan objek file. Setelah digunakan, file tersebut ditutup dengan method close().

Contoh:

```
>>> f = open('/etc/hosts')
>>> f
<open file '/etc/hosts', mode 'r' at 0xb7bf54e8>
>>> f.close()
>>> f
<closed file '/etc/hosts', mode 'r' at 0xb7bf54e8>
>>>
```

Membaca isi file

Untuk membaca isi file yang telah dibuka, beberapa cara bisa digunakan:

- Membaca sekaligus isi file dengan method readlines(). Hasil pembacaan akan disimpan di list.

```
>>> f = open('/tmp/abc')
>>> isi = f.readlines()
>>> f.close()
>>> isi
['test baris 1\n', 'test baris 2\n', 'test baris 3\n']
>>>
```

- Membaca sekaligus isi file dengan method read(). Hasil pembacaan akan disimpan di string.

```
>>> f = open('/tmp/abc')
>>> isi = f.read()
>>> f.close()
>>> isi
'test baris 1\ntest baris 2\ntest baris 3\n'
```

- Membaca baris demi baris dengan method

```
   readline().

>>> f = open('/tmp/abc')
>>> while True:
...     baris = f.readline()
...     if not baris:
...         break
...     print baris
...
test baris 1

test baris 2

test baris 3

>>> f.close()
>>>
```

- Membaca langsung baris tertentu dengan module linecache.

```
>>> import linecache
>>> baris2 = linecache.getline('/tmp/abc',2)
>>> print baris2
test baris 2
```

```
>>> linecache.clearcache()
```

- Membaca sejumlah byte tertentu dengan method read().

```
>>> f = open('/tmp/abc')
>>> buf3 = f.read(3)
>>> print buf3
tes
>>> f.close()
>>>
```

Menulis ke file

Untuk menulis ke file yang telah dibuka, beberapa cara bisa digunakan:

- Menulis string dengan method write.

```
>>> f = open('/tmp/test','w')
>>> f.write('halo apa kabar')
>>> f.close()
>>>
>>> print open('/tmp/test').readlines()
```

```
['halo apa kabar']
```

- Menulis sequence dengan method writelines.

```
>>> lines = ['halo', 'apa', 'kabar']
>>> f = open('/tmp/test2', 'w')
>>> f.writelines(lines)
>>> f.close()
>>>
>>> print open('/tmp/test2').readlines()
['haloapakabar']
>>>
```

27. Lampiran: User interface SQLiteBoy

Apa yang kita bahas berikut mungkin membutuhkan:

- Extended feature diaktifkan
- Login sebagai user level admin

Semua label tombol dan link dalam bahasa default.

Bekerja dengan tabel

Pilihlah terlebih dahulu tabel dari daftar: Table, kemudian lakukanlah langkah berikut.

Lihat isi tabel	Klik tombol browse
Tambahkan data ke dalam tabel	Klik tombol insert
Lihat dan tambahkan kolom	Klik tombol column
Mengganti nama tabel	Klik tombol rename
Mengosongkan isi tabel	Klik tombol empty
Menghapus tabel dan isinya	Klik tombol drop
Mengekspor tabel ke format CSV	Klik tombol export
Menambahkan isi tabel,	Klik tombol import

dengan mengimpor dari file CSV	
Melihat skema tabel dan membuat tabel baru berdasarkan skema tersebut	Klik tombol schema
Mengopi tabel	Klik tombol copy

Untuk membuat tabel, kliklah tombol create.

Bekerja dengan query

Kliklah tombol query.

Vacuum database

Kliklah tombol vacuum.

Bekerja dengan form

Pilihlah terlebih dahulu form dari daftar: Form, kemudian lakukanlah langkah berikut.

Menjalankan form	Klik tombol run
Membuat atau menghapus shortcut	Klik tombol shortcut

Mengedit kode form	Klik tombol edit

Untuk membuat form, kliklah tombol create.

Bekerja dengan report

Pilihlah terlebih dahulu report dari daftar: Report, kemudian lakukanlah langkah berikut.

Menjalankan report	Klik tombol run
Membuat atau menghapus shortcut	Klik tombol shortcut
Mengedit kode report	Klik tombol edit

Untuk membuat report, kliklah tombol create.

Admin

Khusus untuk user level admin. Aktiflah di halaman depan (URL /).

Pengaturan user	Klik link users
Pengaturan hosts	Klik link hosts
Pengaturan sistem	Klik link system
Mendownload file database	Klik link backup
Upload dan menjalankan	Klik link scripts

script	

Source code, readme, website

Aktiflah di halaman depan (URL /).

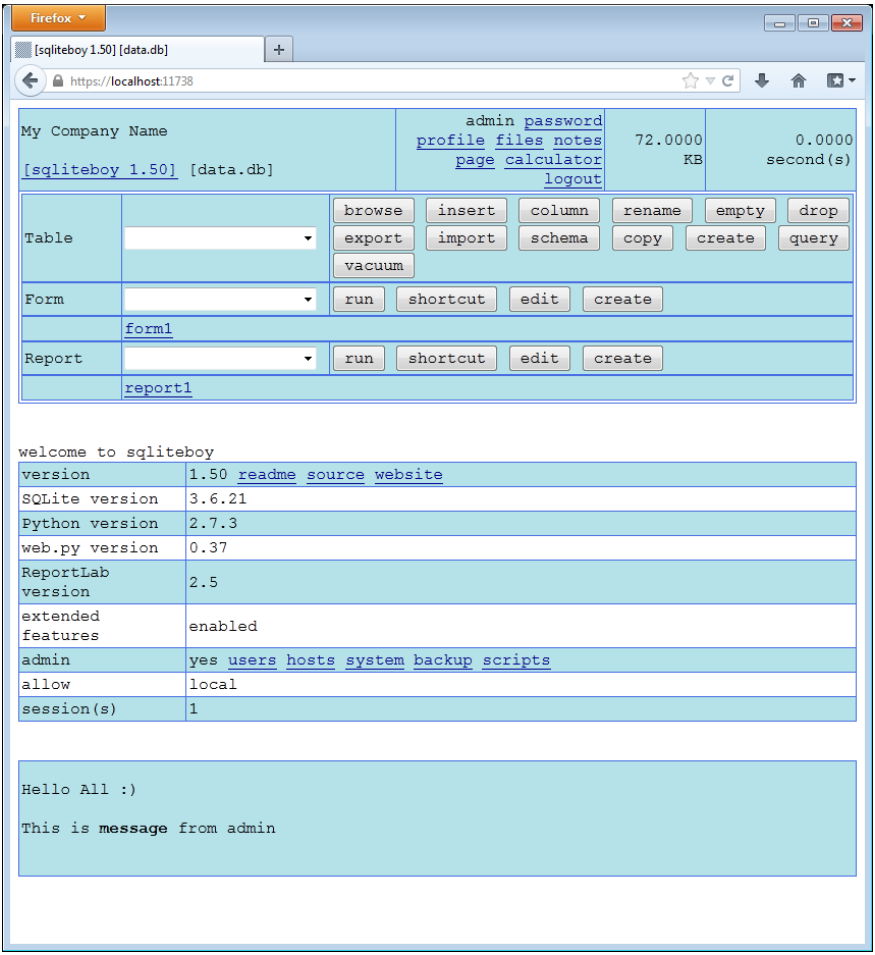
Membaca atau mendownload source code	Klik link source
Membaca atau mendownload readme	Klik link readme
Mengunjungi situs web SqliteBoy	Klik link website

Lain-lain

Semua link berikut dapat ditemukan di sisi atas setiap halaman.

Mengganti password	Klik link password
Melihat atau mengubah profile	Klik link profile
Melihat atau mengupload file	Klik link files
Melihat atau mengedit notes	Klik link notes
Melihat atau mengedit	Klik link page

isi homepage	
Menggunakan kalkulator	Klik link calculator
Logout	Klik link logout


Firefox
[sqliteboy 1.50] [data.db]
https://localhost:11738
My Company Name
[sqliteboy 1.50] [data.db]
admin password profile files notes page calculator logout
72.0000 KB
0.0000 second(s)
Table
browse insert column rename empty drop export import schema copy create query vacuum
Form
run shortcut edit create
form1
Report
run shortcut edit create
report1
welcome to sqliteboy
version 1.50 readme source website
SQLite version 3.6.21
Python version 2.7.3
web.py version 0.37
ReportLab version 2.5
extended features enabled
admin yes users hosts system backup scripts
allow local
session(s) 1
Hello All :)
This is message from admin