BAB 2

LANDASAN TEORI

2.1 Pengertian Android

Android adalah sebuah kumpulan perangkat lunak untuk perangkat *mobile* yang mencakup sistem operasi, *middleware* dan aplikasi utama *mobile.* Android memiliki empat karakteristik sebagai berikut:

1. Terbuka

   Android dibangun untuk benar-benar terbuka sehingga sebuah aplikasi dapat memanggil salah satu fungsi inti ponsel seperti membuat panggilan, mengirim pesan teks, menggunakan kamera, dan lain-lain. Android menggunakan sebuah mesin virtual yang dirancang khusus untuk mengoptimalkan sumber daya memori dan perangkat keras yang terdapat di dalam perangkat. Android merupakan *open source*, dapat secara bebas diperluas untuk memasukkan teknologi baru yang lebih maju pada saat teknologi tersebut muncul. *Platform* ini akan terus berkembang untuk membangun aplikasi *mobile* yang inovatif.

2. Semua aplikasi dibuat sama

Android tidak memberikan perbedaan terhadap aplikasi utama dari telepon dan aplikasi pihak ketiga (*third-party application*). Semua aplikasi dapat dibangun untuk memiliki akses yang sama terhadap kemampuan sebuah telepon dalam menyediakan layanan dan aplikasi yang luas terhadap para pengguna.

3. Memecahkan hambatan pada aplikasi

Android memecah hambatan untuk membangun aplikasi yang baru dan inovatif. Misalnya, pengembang dapat menggabungkan informasi yang diperoleh dari *web* dengan data pada ponsel seseorang seperti kontak pengguna, kalender, atau lokasi geografis.

4. Pengembangan aplikasi yang cepat dan mudah

Android menyediakan akses yang sangat luas kepada pengguna untuk menggunakan *library* yang diperlukan dan *tools* yang dapat digunakan untuk membangun aplikasi yang semakin baik. Android memiliki sekumpulan *tools* yang dapat digunakan sehingga membantu para pengembang dalam meningkatkan produktivitas pada saat membangun aplikasi yang dibuat.(*Sumber :* http://www.android.com/about/)

Google Inc. sepenuhnya membangun Android dan menjadikannya bersifat terbuka (*open source*) sehingga para pengembang dapat menggunakan Android tanpa mengeluarkan biaya untuk lisensi dari Google

dan dapat membangun Android tanpa adanya batasan-batasan. Android *Software Development Kit* (SDK) menyediakan alat dan *Application Programming Interface* (API) yang diperlukan untuk mulai mengembangkan aplikasi pada *platform* Android menggunakan bahasa pemrograman Java.

2.1.1 Sejarah Sistem Operasi Android

Telepon seluler menggunakan berbagai macam sistem operasi seperti *Symbian OS*®, *Microsoft's Windows Mobile*®, *Mobile Linux*®, *iPhone OS*® (berdasarkan *Mac OS X*), *Moblin*® (dari Intel), dan berbagai macam sistem operasi lainnya. API yang tersedia untuk mengembangkan aplikasi *mobile* terbatas dan oleh karena itulah Google mulai mengembangkan dirinya. *Platform* Android menjanjikan keterbukaan, kemudahan untuk menjangkau, *source code* yang terbuka, dan pengembangan *framework* yang *high end*.

Google membeli perusahaan Android Inc., yang merupakan sebuah perusahaan kecil berbasis pengembangan perangkat lunak untuk ponsel, pada tahun 2005 untuk memulai pengembangan pada *platform* Android. Tokoh utama pada Android Inc. meliputi Andy Rubin, Rich Miner, Nick Sears, dan Chris White.

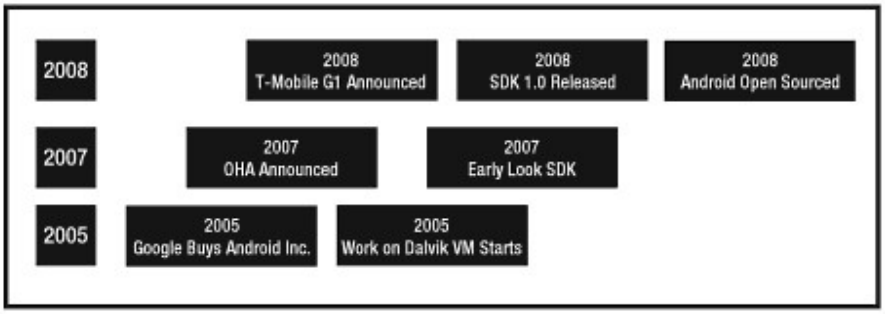
Pada tanggal 5 November 2007, kelompok pemimpin industri bersama-sama membentuk *Open Handset Alliance* (OHA) yang diciptakan untuk mengembangkan standar terbuka bagi perangkat *mobile*. OHA terdiri dari 34 anggota besar dan

beberapa anggota yang terkemuka diantaranya sebagai berikut: Sprint Nextel®, T-Mobile®, Motorola®, Samsung®, Sony Ericsson®, Toshiba®, Vodafone®, Google, Intel® dan Texas Instruments.

Android SDK dirilis pertama kali pada 12 November 2007 dan para pengembang memiliki kesempatan untuk memberikan umpan balik dari pengembangan SDK tersebut. Pada bulan September 2008, T-Mobile memperkenalkan ketersediaan T-Mobile G1 yang merupakan *smart phone* pertama berbasiskan *platform* Android. Beberapa hari kemudian, Google merilis Android SDK 1.0. Google membuat *source code* dari *platform* Android menjadi tersedia di bawah lisensi *Apache's open source*.

Google merilis perangkat genggam (disebut Android Dev Phone 1) yang dapat menjalankan aplikasi Android tanpa terikat oleh berbagai jaringan *provider* telepon seluler pada akhir 2008. Tujuan dari perangkat ini adalah memungkinkan pengembang untuk melakukan percobaan dengan perangkat sebenarnya yang dapat menjalankan Android OS tanpa berbagai kontrak. Google juga merilis versi 1.1 dari sistem operasi Android pada waktu yang tidak lama. Versi 1.1 dari Android tidak mendukung adanya *soft keyboards* dan membutuhkan perangkat yang memiliki *keyboard* secara fisik. Android menyelesaikan masalah ini dengan merilis versi 1.5 pada bulan April 2009 dengan sejumlah tambahan fitur seperti kemampuan perekaman media,*widgets*, dan *live folders*.

Versi 1.6 dari Android OS dirilis pada bulan September 2009 dan hanya dalam waktu satu bulan versi Android 2.0 dirilis dan membanjiri seluruh perangkat Android. Versi ini memiliki kemampuan *advanced search*, *text to speech*, *gestures*, dan *multi touch*. Android 2.0 memperkenalkan kemampuan untuk menggunakan HTML karena didukung oleh HTML 5. Semakin banyak aplikasi berbasiskan Android setiap harinya yang terdapat pada *application store* secara *online* atau dikenal sebagai *Android Market*.



Gambar 2.1 Gambar Sejarah Sistem Operasi Android

2.1.2 Fitur Sistem Operasi Android

Sistem operasi Android memiliki fitur-fitur sebagai berikut:

1. Kerangka kerja aplikasi (application framework)

   Digunakan untuk menulis aplikasi di Android sehingga memungkinkan penggunaan kembali dan penggantian komponen. Kerangka kerja ini didukung oleh berbagai open source libraries seperti openssl, sqlite,dan libc serta didukung oleh libraries utama Android. Kerangka kerja sistem operasi Android didasarkan pada UNIX file system permission yang menjamin bahwa

aplikasi-aplikasi tersebut hanya memiliki kemampuan yang diberikan oleh pemilik ponsel pada waktu penginstalan.

2. *Dalvik Virtual Machine* (DVM)

Dalvik Virtual Machine (DVM) adalah sebuah mesin virtual yang menggunakan memori yang sangat rendah dan secara khusus dirancang untuk Android untuk dijalankan pada *embedded system*. DVM bekerja dengan baik pada situasi dengan tenaga yang rendah dan mengoptimalkan perangkat *mobile*. DVM juga mengatur atribut dari *Central Processing Unit* (CPU) serta membuat sebuah format *file* yang spesial (.DEX) yang dibuat selama *build time post processing*. DVM mengambil *file* yang dihasilkan oleh *class* Java dan menggabungkannya ke dalam satu atau lebih *Dalvik Executable* (.dex). DVM dapat menggunakan kembali salinan informasi dari beberapa *class file* dan secara efektif mengurangi kebutuhan penyimpanan oleh setengah dari *Java Archive* (.jar) *file* tradisional. Konversi antara kelas Java dan format (.dex) dilakukan dengan memasukkan "dx tool".

DVM menggunakan *assembly-code* yang berbeda dimana DVM menggunakan *register* sebagai unit utama dari penyimpanan data daripada menggunakan *stack*. Hasil akhir dari *executable-code* pada Android, merupakan hasil dari DVM yang didasarkan bukan pada Java *byte-code* melainkan pada file (.dex). Hal ini berarti bahwa Java *byte-code* tidak dieksekusi secara langsung melainkan dimulai dari Java *classfile* terlebih

dahulu dan kemudian mengkonversikannya ke dalam *file* (.dex) yang berhubungan.

a. Browser yang terintegrasi
b. Grafik yang teroptimasi
c. SQLite
d. Dukungan media untuk suara, video dan format
e. GSM *Telephony* (bergantung dari perangkat keras yang digunakan)
f. *Bluetooth*, *Enhanced Data rates for GSM Evolution* (EDGE), *3rd Generation* (3G), dan WiFi™ (bergantung dari perangkat keras yang digunakan)
g. Kamera, *Global Postioning System* (GPS), kompas dan accelerometer (bergantung dari perangkat keras yang digunakan).
h. Lingkungan pengembangan yang lengkap, seperti emulator, peralatan untuk *debugging*, memori dan *performance. profiling*, serta *plug-in* untuk Eclipse IDE.

(*Sumber :* http://www.scribd.com/doc/7577184/Android)

2.1.3 Arsitektur Sistem Operasi Android

Sistem Operasi Android memiliki komponen utama sebagai berikut :

a. Aplikasi

Android berisi sekumpulan aplikasi utama seperti : *email client*, program *Short Message Service* (SMS), kalender, peta, browser, daftar kontak, dan

lain-lain. Semua aplikasi ditulis dengan menggunakan bahasa pemgrograman Java.

b. Kerangka kerja aplikasi

Kerangka kerja aplikasi yang ditulis dengan menggunakan bahasa pemrograman Java merupakan peralatan yang digunakan oleh semua aplikasi, baik aplikasi bawaan dari ponsel seperti daftar kontak, dan kotak SMS, maupun aplikasi yang ditulis oleh Google ataupun pengembang Android.

Android menawarkan para pengembang kemampuan untuk membangun aplikasi yang inovatif. Pengembang bebas untuk mengambil keuntungan dari perangkat keras, akses lokasi informasi, menjalankan *background services*, mengatur alarm, menambahkan peringatan ke status bar, dan masih banyak lagi. Pengembang memiliki akses yang penuh ke dalam kerangka kerja API yang sama yang digunakan oleh aplikasi utama. Pada dasarnya, kerangka kerja aplikasi memiliki beberapa komponen sebagai berikut:

a. *Activity Manager*

Mengatur siklus dari aplikasi dan menyediakan navigasi *backstack* untuk aplikasi yang berjalan pada proses yang berbeda.

b. *Package Manager*

Untuk melacak aplikasi yang di-*instal* pada perangkat.

c. *Windows Manager*

Merupakan abstraksi dari bahasa pemrograman Java pada bagian atas dari level *services* (pada level yang lebih rendah) yang disediakan oleh *Surface Manager*.

d. *Telephony Manager*

Berisi sekumpulan API yang diperlukan untuk memanggil aplikasi.

e. *Content Providers*

Digunakan untuk memungkinkan aplikasi mengakses data dari aplikasi lain (seperti *contacts*) atau untuk membagikan data mereka sendiri.

f. *Resource Manager*

Digunakan untuk menagkses sumber daya yang bersifat bukan *code* seperti *string* lokal, *bitmap*, deskripsi dari *layout file* dan bagian eksternal lain dari aplikasi.

g. *View System*

Digunakan untuk mengambil sekumpulan *button*, *list, grid,* dan *text box* yang digunakan di dalam antarmuka pengguna.

h. *Notification Manager*

Digunakan untuk mengatur tampilan peringatan dan fungsi-fungsi lain.

c. *Libraries*

Android memiliki sekumpulan *library* C/C++ yang digunakan oleh berbagai komponen dalam sistem Android. Kemampuan-kemampuan ini dilihat oleh para pengembang melalui kerangka kerja aplikasi.

Beberapa dari *library* utama dijelaskan sebagai berikut:

a. *System C Library*

Merupakan implementasi turunan dari standar *system library* C (libc) yang diatur untuk peralatan berbasis *embedded* Linux.

b. *Media Libraries*

Disediakan oleh PacketVideo (salah satu anggota dari OHA) yang memberikan *library* untuk memutar ulang dan menyimpan format suara dan video, serta *static image file* seperti MPEG4, MP3, AAC, AMR, JPG, and PNG.

c. *Surface Manager*

Mengatur akses ke dalam subsistem tampilan dan susunan grafis *layer* 2D dan 3D secara mulus dari beberapa aplikasi dan menyusun permukaan gambar yang berbeda pada layar ponsel.

d. *LibWebCore*

Merupakan *web browser* modern yang menjadi kekuatan bagi *browser* Android dan sebuah *embeddable web view*.

e. *Scalable Graphics Library* (SGL)

SGL mendasari mesin grafis 2D dan bekerja bersama-sama dengan lapisan pada level yang lebih tinggi dari kerangka kerja (seperti *Windows Manager* dan *Surface Manager*) untuk mengimplementasikan keseluruhan *graphics pipeline* dari Android.

f. *3DLibraries*

Implementasi yang didasarkan pada OpenGL ES 1.0 APIs dimana *library* menggunakan baik akselerasi perangkat keras 3D (jika tersedia) ataupun yang disertakan, dengan rasterisasi perangkat lunak 3D yang sangat optimal.

g. *FreeType Library*

Digunakan untuk menghaluskan semua tulisan *bitmap* dan vektor.

h. SQLite

Merupakan *relational database* yang kuat dan ringan serta tersedia untuk semua aplikasi.

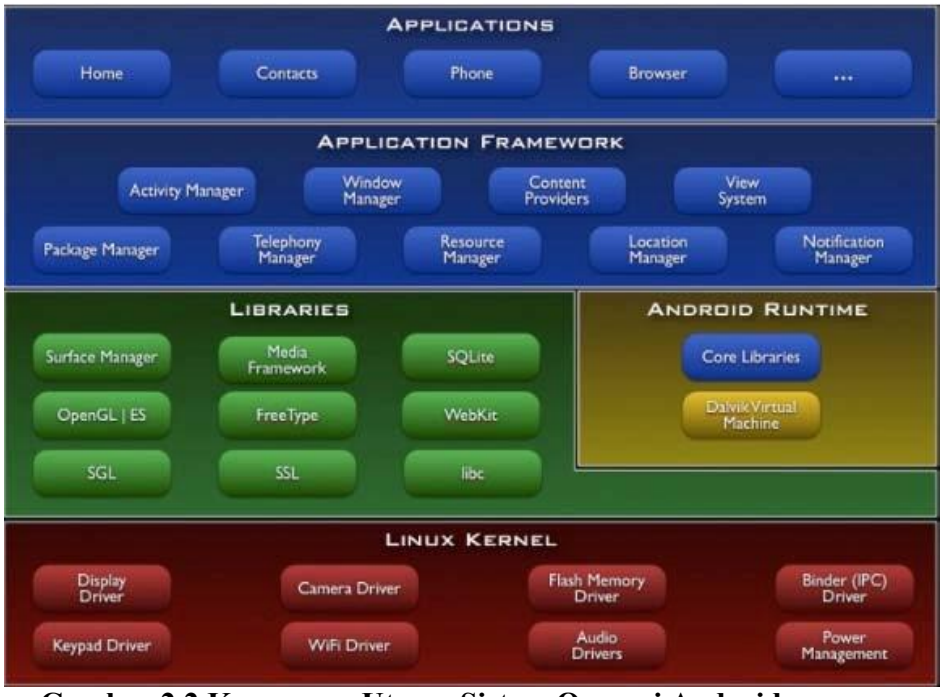
d. *Android Runtime*

Merupakan lokasi dimana komponen utama dari DVM ditempatkan. DVM dirancang secara khusus untuk Android pada saat dijalankan pada lingkungan yang terbatas, dimana baterai yang terbatas, CPU, memori, dan penyimpanan data menjadi fokus utama. Android memiliki sebuah *tool* yang terintegrasi yaitu "dx" yang mengkonversi *generated byte code* dari (.JAR) ke dalam file (.DEX) sehingga *byte code* menjadi lebih efisien untuk dijalankan pada prosesor yang kecil. Hal ini memungkinkan untuk memiliki beberapa jenis dari DVM berjalan

pada suatu peralatan tunggal pada waktu yang sama. *Core libraries* ditulis dalam bahasa Java dan berisi kumpulan *class*, I/O dan peralatan lain.

e. *Linux Kernel*

Arsitektur Android berdasarkan pada Linux 2.6 kernel yang dapat digunakan untuk mengatur keamanan, manajemen memori, manajemen proses, *network stack*, dan *driver model*. Kernel juga bertindak sebagai lapisan abstrak antara perangkat keras dan seluruh *software stack*. Diagram di bawah ini menunjukkan komponen utama dari sistem operasi Android:



Gambar 2.2 Komponen Utama Sistem Operasi Android
(*Sumber :* http://developer.android.com/guide/basics/what-is-android.html)

2.1.4 Versi Android

Android memiliki sejumlah pembaharuan semenjak rilis aslinya. Pembaharuan ini dilakukan untuk memperbaiki *bug* dan menambah fitur-fitur yang baru. Berikut merupakan versi-versi yang dimiliki Android sampai saat ini:

Tabel 2.1 Versi - versi Android

Versi Android	API Level	Nickname
Android 1.0	1	
Android 1.1	2	
Android 1.5	3	Cupcake
Android 1.6	4	Donut
Android 2.0	5	Eclair
Android 2.01	6	Eclair
Android 2.1	7	Eclair
Android 2.2	8	Froyo (Frozen Yogurt)
Android 2.3	9	Gingerbread
Android 2.3.3	10	Gingerbread
Android 3.0	11	Honeycomb
Android 3.1	12	Honeycomb
Android 3.2	13	Honeycomb
Android 4.0	14	Ice Cream Sandwich
Android 4.0.3	15	Ice Cream Sandwich
Android 4.1	16	Jelly Bean

2.2 *Database Management System* (DBMS)

Menurut Connolly dan Begg (2002), *Database Management System* merupakan sebuah sistem perangkat lunak yang memungkinkan pengguna untuk mendefinisikan, membuat, mengatur, dan mengontrol pengaksesan ke dalam basis data.Sebuah DBMS menyediakan fasilitas– fasilitas sebagai berikut :

1. DBMS memungkinkan pengguna untuk mendefinisikan basis data, dengan menggunakan *Data Definition Language* (DDL). DDL memungkinkan pemakai untuk menspesifikasikan tipe – tipe dan struktur data serta *constraint* pada data untuk disimpan di dalam basis data.
2. DBMS memungkinkan pengguna untuk memasukkan, mengubah, menghapus, dan mengambil data dari basis data dengan menggunakan *Data Manipulation Language* (DML).
3. DBMS menyediakan pengontrolan akses ke dalam basis data. Contohnya, DBMS menyediakan :
    a. *Security System,* mencegah pengguna yang tidak memiliki hak dalam mengakses basis data
    b. *Integrity system,* mengatur konsistensi dari data yang disimpan.
    c. *Concurrency control system,* memungkinkan pengaksesan basis data secara bersama – sama.
    d. *Recovery control system,* memperbaiki basis data kembali ke bentuk semula sebelum terjadinya kerusakan perangkat keras atau kerusakan perangkat lunak.

e. *User-accessible catalog,* berisi gambaran dari data yang terdapat di dalam basis data.

2.3 Model Proses *Waterfall*

Menurut Pressman (2001, p.28), sekuensial linier atau lebih yang dikenal dengan sebutan *waterfall* model menawarkan sebuah pendekatan yang sistematis dan sekuensial dalam pengembangan piranti lunak yang dimulai dari label sistem dan perkembangannya dengan melalui tahap analisis, desain, *coding* dan *testing*.

Model *waterfall* adalah versi paling popular dari s*ystem development lifecycle model* untuk *software engineering*. Model ini sering dianggap sebagai pendekatan klasik dalam daur hidup pengembangan sistem. Ada empat tahapan utama dalam model ini, yaitu:

1. Analisis

   Analisis adalah sebuah proses pengumpulan kebutuhan yang dikhususkan dan difokuskan dalam pembuatan piranti lunak.

2. Desain

   Desain adalah sebuah proses yang menerjemahkan hasil dari analisis dalam bentuk representasi piranti lunak sehingga dapat dinilai kualitasnya sebelum proses *coding* dimulai.

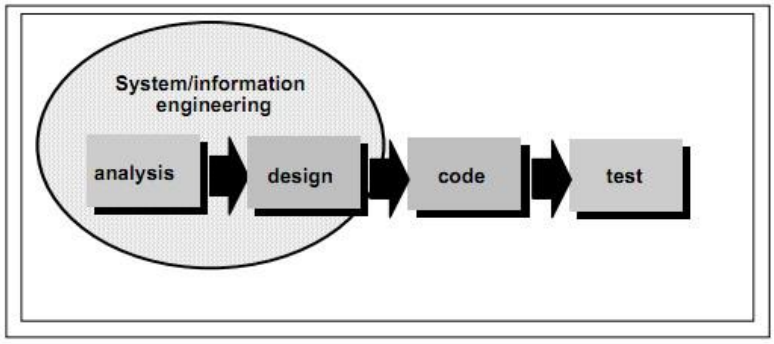
3. *Coding*

Coding adalah proses dimana hasil dari desain diterjemahkan kembali dalam bentuk bahasa pemrograman yang dapat dimengerti oleh mesin.

4. *Testing*

Testing dilakukan setelah *coding* selesai dilakukan. Testing digunakan untuk mengetahui kesalahan yang tidak terdeteksi sebelumnya atau hasil dari proses yang tidak diinginkan.

Adapun gambaran dari metodologi waterfall adalah sebagai berikut:



Gambar 2.3 Model Metodologi *Waterfall*

(*Sumber : Pressman, Software Engineering, 2001*)

2.4 Database SQLite

Menurut Jay A. Kreibich (2010,12) SQLite merupakan paket perangkat lunak yang bersifat *public domain* yang menyediakan sistem manajemen basis data relasional atau RDBMS. Sistem basis data relasional digunakan untuk menyimpan *record* yang didefinisikan oleh pengguna pada ukuran tabel yang besar dan memproses perintah

query yang kompleks dan menggabungkan data dari berbagai tabel untuk menghasilkan laporan dan rangkuman data.

Kata 'Lite' pada SQLite tidak menunjuk pada kemampuannya, melainkan menunjuk pada sifat dari SQLite, yaitu ringan ketika dihubungkan dengan kompleksitas pengaturan, *administrative overhead*, dan pemakaian sumber.

SQLite memiliki fitur-fitur sebagai berikut :

1. Tidak memerlukan *server*

   Arsitektur SQLite tidak memiliki arsitektur *client server*. Kebanyakan sistem database skala besar memiliki paket *server* yang besar yang membentuk mesin *database*.

2. *Single File Database*

   SQLite mengemas seluruh *database* ke dalam suatu *single file*. *Single file* tersebut berisi *layoutdatabase* dan data aktual yang berada pada tabel dan indeks yang berbeda. Format *file* dapat digunakan pada banyak *platform* dan dapat diakses pada mesin manapun tanpa memperhatikan *native byte order* ataupun ukuran kata.

   Pengemasan database kedalam suatu *file* tunggal memudahkan pengguna untuk membuat, menyalin, ataupun mem-*backupimage database* yang berada di dalam media penyimpanan

3. *Zero Configuration*

   SQLite tidak membutuhkan apapun untuk melakukan instalasi dan konfigurasi. Dengan mengeliminasi *server* dan menggabungkan *database*

secara langsung ke dalam aplikasi, maka pengguna tidak perlu mengetahui bahwa mereka sedang menggunakan *database*.

4. *Embedded Device Support*

   Ukuran *code* dari SQLite bersifat kecil dan penggunaan sumber daya yang konservatif membuatnya cocok digunakan untuk *embedded system* yang berjalan terbatas pada sistem operasi.

5. Fitur-fitur yang unik

   SQLite menggunakan sistem dengan tipe dinamis untuk tabel-tabel. SQLite memungkinkan pengguna untuk memasukkan nilai ke dalam kolom tanpa memperhatikan tipe data. Pada beberapa cara pemakaiannya, sistem yang bertipe dinamis pada SQLite mirip dengan sistem yang ditemukan pada bahasa *scripting* yang populer, yang sering memiliki sebuah tipe skalar yang dapat menerima semua tipe data dari integer sampai string. Fitur lainnya adalah kemampuan untuk memanipulasi lebih dari satu basis data pada satu waktu. SQLite mempunyai kemampuan dalam menghubungkan sebuah koneksi *database* tunggal dengan banyak file basis data secara bersamaan. Hal ini memungkinkan SQLite umtuk memproses SQL *statement* yang menjembatani beberapa basis data sekaligus.

6. *Compatible License*

   SQLite dan SQLite *code* tidak memiliki lisensi pengguna dan tidak dilindungi oleh *GNU's Not Unix* (GNU) *General Public License* (GPL) atau lisensi *open source* sejenisnya. Hal ini berarti pengguna dapat melakukan apapun dengan *source code* SQLite, sehingga *library code* dapat digunakan dengan berbagai

cara, dimodifikasi dengan berbagai cara dan didistribusikan dengan berbagai cara.

7. *Highly reliable*

   Sejumlah tes telah dilakukan sebelum *library* SQLite masing-masing dirilis. Hal ini dilakukan untuk mempertahankan tingkat kehandalan yang tinggi.

Bahasa SQL (*Structured Query Language*)

Bahasa SQL merupakan sarana utama berinteraksi dengan hampir semua sistem basis data relasional modern. SQL menyediakan perintah untuk mengkonfigurasi tabel, indeks, dan struktur data dalam basis data. Perintah SQL juga digunakan untuk memasukkan, mengedit, dan menghapus data, serta *query* untuk mencari nilai suatu data yang spesifik.

Semua interaksi dengan basis data relasional dilakukan melalui bahasa SQL. Hal ini berlaku ketika mengetik perintah secara interaktif atau saat menggunakan pemrograman API.

Perintah SQL dibagi menjadi empat kategori utama. Setiap kategori mendefinisikan sekumpulan perintah yang memiliki tujuan masing-masing. Kategori-kateogri tersebut adalah sebagai berikut:

1. *Data Definition Language* (DDL)

   Mengacu pada perintah yang menentukan struktur tabel, *view* dan indeks atau perintah yang digunakan untuk menjelaskan objek dari basis data.

Contoh dari perintah DDL adalah :

1. CREATE TABLE (digunakan untuk mendefinisikan sebuah tabel baru)
2. CREATE DATABASE (digunakan untuk mendefinisikan sebuah basis data yang baru)
3. CREATE INDEX (digunakan untuk membuat *index*)
4. CREATE VIEW (digunakan untuk membuat *view*)
5. DROP DATABASE (digunakan untuk menghapus basis data)
6. DROP TABLE (digunakan untuk menghapus tabel)
7. DROP INDEX (digunakan untuk menghapus index)
8. DROP VIEW (digunakan untuk menghapus view)
9. ALTER TABLE (digunakan untuk mengubah suatu tabel)

2. *Data Manipulation Language* (DML)

Mengacu pada semua perintah untuk menambahkan, mengedit, menghapus dan mengambil nilai dari data aktual yang didefinisikan oleh DDL.

Contoh dari perintah DML adalah:

1. INSERT (digunakan untuk memasukkan nilai baru ke dalam tabel)
2. UPDATE (digunakan untuk mengubah suatu nilai pada satu atau lebih kolom)
3. DELETE (digunakan untuk menghapus suatu nilai pada tabel)
4. SELECT (digunakan untuk mengambil data dari tabel)

3. *Transaction Control Language* (TCL)

Perintah TCL dapat digunakan untuk mengontrol perintah dari transaksi DML dan DDL.

Contoh dari perintah TCL adalah sebagai berikut:

1. BEGIN (digunakan untuk memulai transaksi *multistatement*)
2. COMMIT (digunakan untuk mengakhiri dan menerima transaksi)
3. ROLLBACK (digunakan untuk membatalkan perubahan yang dilakukan dalam *transaction* atau menghapus semua data yang dimodifikasi pada awal *transaction*)

4. *Data Control Language* (DCL)

Tujuan utama dari DCL adalah untuk memberikan atau mencabut kontrol akses. Sama seperti hak akses file, perintah DCL digunakan untuk memungkinkan (atau menolak) izin pengguna basis data tertentu (atau kelompok pengguna) untuk menggunakan atau mengakses sumber daya tertentu di dalam basis data. Izin ini dapat diterapkan baik pada DDL (seperti membuat tabel) maupun DML (seperti membaca, mengedit, dan menghapus *record* pada tabel yang spesifik)

Contoh dari perintah TCL adalah:

1. GRANT (digunakan untuk memberikan izin)
2. REVOKE (digunakan untuk menghapus izin yang sudah ada)

SQLite mendukung perintah standar DDL, DML, dan TCL tetapi tidak memiliki perintah DCL. SQLite tidak memiliki konsep untuk meminta izin

atas hak akses karena SQLite tidak memiliki *username* atau *login*. SQLite bergantung pada hak akses dari tipe data untuk mendefinisikan siapa saja yang dapat membuka dan mengakses basis data.