M. Bahrul Ulum, S.Kom, M.Kom





Pemrograman Mobile

**Firebase Realtime Database**

**Firebase Realtime Database**

Firebase Realtime Database adalah database yang di-host di cloud. Data disimpan sebagai JSON dan disinkronkan secara realtime ke setiap klien yang terhubung. Ketika Anda membuat aplikasi lintas-platform dengan SDK Android, iOS, dan JavaScript, semua klien akan berbagi sebuah instance Realtime Database dan menerima update data terbaru secara otomatis.

**Kemampuan utama**

|  |  |
| --- | --- |
| Realtime | Sebagai ganti permintaan HTTP biasa, Firebase Realtime Database menggunakan sinkronisasi data—setiap kali data berubah, semua perangkat yang terhubung akan menerima update dalam waktu milidetik. Memberikan pengalaman yang kolaboratif dan imersif tanpa perlu memikirkan kode jaringan.  |
| Offline | Aplikasi Firebase tetap responsif bahkan saat offline karena SDK Firebase Realtime Database menyimpan data ke disk. Setelah konektivitas pulih, perangkat klien akan menerima setiap perubahan yang terlewat dan melakukan sinkronisasi dengan status server saat ini.  |
| Dapat Diakses dari Perangkat Klien | Firebase Realtime Database dapat diakses secara langsung dari perangkat seluler atau browser web; server aplikasi tidak diperlukan. Keamanan dan validasi data dapat diakses melalui Aturan Keamanan Firebase Realtime Database yang merupakan kumpulan aturan berbasis ekspresi dan dijalankan ketika data dibaca atau ditulis.  |
| Menskalakan di beberapa database | Dengan Firebase Realtime Database pada paket harga Blaze, Anda dapat mendukung kebutuhan data aplikasi Anda pada skala tertentu dengan membagi data Anda di beberapa instance database di project Firebase yang sama. Menyederhanakan autentikasi dengan Firebase Authentication pada project Anda dan mengautentikasi pengguna di instance database Anda. Mengontrol akses ke data di tiap database dengan Aturan Firebase Realtime Database khusus untuk tiap instance database.  |

**Bagaimana cara kerjanya?**

Firebase Realtime Database memungkinkan Anda untuk membuat aplikasi kolaboratif dan kaya fitur dengan menyediakan akses yang aman ke database, langsung dari kode sisi klien. Data disimpan di drive lokal. Bahkan saat offline sekalipun, peristiwa realtime terus berlangsung, sehingga pengguna akhir akan merasakan pengalaman yang responsif. Ketika koneksi perangkat pulih kembali, Realtime Database akan menyinkronkan perubahan data lokal dengan update jarak jauh yang terjadi selama klien offline, sehingga setiap perbedaan akan otomatis digabungkan.

Realtime Database menyediakan bahasa aturan berbasis ekspresi yang fleksibel, atau disebut juga Aturan Keamanan Firebase Realtime Database, untuk menentukan metode strukturisasi data dan kapan data dapat dibaca atau ditulis. Ketika diintegrasikan dengan Firebase Authentication, developer dapat menentukan siapa yang memiliki akses ke data tertentu dan bagaimana mereka dapat mengaksesnya.

Realtime Database adalah database NoSQL, sehingga memiliki pengoptimalan dan fungsionalitas yang berbeda dengan database terkait. API Realtime Database dirancang agar hanya mengizinkan operasi yang dapat dijalankan dengan cepat. Hal ini memungkinkan Anda untuk membangun pengalaman realtime yang luar biasa dan dapat melayani jutaan pengguna tanpa mengorbankan kemampuan respons. Oleh karena itu, perlu dipikirkan bagaimana pengguna mengakses data, kemudian buat struktur data sesuai dengan kebutuhan tersebut.

**Alur implementasi**

|  |  |  |
| --- | --- | --- |
|  | Mengintegrasikan Firebase Realtime Database SDK | Sertakan klien dengan cepat melalui Gradle, CocoaPods, atau skrip. |
|  | Membuat Referensi Realtime Database | Referensikan data JSON Anda, seperti "users/user:1234/phone\_number", untuk menetapkan data atau berlangganan perubahan data.  |
|  | Menetapkan Data dan Mendeteksi Perubahan | Gunakan referensi ini untuk menuliskan data atau berlangganan perubahan. |
|  | Mengaktifkan Persistensi Offline | Izinkan penulisan data ke disk lokal perangkat agar tetap tersedia saat offline.  |
|  | Melindungi data | Gunakan Aturan Keamanan Firebase Realtime Database untuk melindungi data Anda. |

## Ingin menyimpan jenis data lain?

* [Firebase Remote Config](https://firebase.google.com/docs/remote-config?hl=id) menyimpan key-value pair yang ditentukan developer untuk mengubah perilaku dan penampilan aplikasi tanpa mewajibkan pengguna untuk mendownload update.
* [Firebase Hosting](https://firebase.google.com/docs/hosting?hl=id) menghosting HTML, CSS, dan JavaScript untuk situs Anda, serta aset lain yang disediakan developer, seperti gambar, font, dan ikon.
* [Cloud Storage](https://firebase.google.com/docs/storage?hl=id) menyimpan file seperti gambar, video, dan audio serta konten buatan pengguna lainnya.

## Langkah berikutnya:

* Tetapkan data dan deteksi perubahan dengan menggunakan SDK [iOS](https://firebase.google.com/docs/database/ios/start?hl=id), [Android](https://firebase.google.com/docs/database/android/start?hl=id), [Web](https://firebase.google.com/docs/database/web/start?hl=id), [Admin](https://firebase.google.com/docs/database/admin/start?hl=id), atau [REST API](https://firebase.google.com/docs/database/rest/start?hl=id).
* Tambahkan Firebase Realtime Database ke aplikasi [iOS](https://firebase.google.com/docs/database/ios/start?hl=id), [Android](https://firebase.google.com/docs/database/android/start?hl=id), atau [Web](https://firebase.google.com/docs/database/web/start?hl=id) Anda.
* Pelajari cara melindungi file menggunakan [Aturan Firebase Realtime Database](https://firebase.google.com/docs/database/security/quickstart?hl=id).
* Jika Anda melakukan upgrade dari SDK Firebase 2.x, lihat panduan upgrade untuk [Android/Java](https://firebase.google.com/support/guides/firebase-android?hl=id), [iOS](https://firebase.google.com/support/guides/firebase-ios?hl=id), atau [Web/Node.js](https://firebase.google.com/support/guides/firebase-web?hl=id).

# Memilih Database: Cloud Firestore atau Realtime Database

Firebase menawarkan dua solusi database berbasis cloud yang dapat diakses klien, yang mendukung sinkronisasi data secara real-time:

* **Realtime Database** adalah database asli Firebase. Solusi ini berlatensi rendah dan efisien untuk aplikasi seluler yang membutuhkan status sinkronisasi di seluruh klien secara real-time.
* **Cloud Firestore** adalah database unggulan baru dari Firebase untuk pengembangan aplikasi seluler. Cloud Firestore melanjutkan keberhasilan Realtime Database dengan model data baru yang lebih intuitif. Cloud Firestore juga memiliki fitur kueri yang lebih kaya dan lebih cepat, dan skala yang lebih baik daripada Realtime Database.

## Database mana yang tepat untuk project Anda?

Kami merekomendasikan Cloud Firestore untuk sebagian besar developer yang memulai project baru. Cloud Firestore menawarkan fungsi, performa, dan skalabilitas tambahan pada infrastruktur yang dirancang untuk mendukung fitur yang lebih canggih dalam rilis mendatang. Anda akan melihat jenis kueri baru, aturan keamanan yang lebih kuat, dan peningkatan performa, selain fitur-fitur lanjutan lainnya yang direncanakan untuk Cloud Firestore.

Saat akan memilih solusi database, pertimbangkan perbedaan antara Cloud Firestore dan Realtime Database berikut ini.

### Model data

Baik Realtime Database dan Cloud Firestore adalah Database NoSQL.

|  |  |
| --- | --- |
| **Realtime Database** | **Cloud Firestore** |
| **Menyimpan data sebagai satu hierarki JSON yang besar.** * Data sederhana sangat mudah disimpan.
* Data kompleks dan hierarkis lebih sulit disusun dalam skala besar.

Pelajari lebih lanjut [model data Realtime Database](https://firebase.google.com/docs/database/web/structure-data?hl=id). | **Menyimpan data dalam dokumen yang tersusun dalam koleksi.** * Data sederhana mudah disimpan dalam dokumen, yang sangat mirip dengan JSON.
* Data kompleks dan hierarkis lebih mudah disusun dalam skala besar, dengan menggunakan subkoleksi dalam dokumen.
* Membutuhkan sedikit denormalisasi dan perataan data.

Pelajari lebih lanjut [model data Cloud Firestore](https://firebase.google.com/docs/firestore/data-model?hl=id). |

### Dukungan real-time dan offline

Keduanya memiliki SDK real-time dan memprioritaskan aplikasi seluler, dan keduanya mendukung penyimpanan data lokal untuk aplikasi siap offline.

|  |  |
| --- | --- |
| **Realtime Database** | **Cloud Firestore** |
| **Dukungan offline untuk klien seluler hanya di iOS dan Android.** | **Dukungan offline untuk klien iOS, Android, dan web.** |

### Membuat kueri

Mengambil, mengurutkan, dan memfilter data dari salah satu database melalui kueri.

|  |  |
| --- | --- |
| **Realtime Database** | **Cloud Firestore** |
| **Kueri mendalam dengan** [**fungsi pengurutan dan pemfilteran**](https://firebase.google.com/docs/database/web/lists-of-data?hl=id#sorting_and_filtering_data) **terbatas.** * Anda hanya dapat mengurutkan atau memfilter properti, bukan mengurutkan dan memfilter properti, dalam satu kueri.
* Kueri bersifat mendalam secara default. Kueri selalu menghasilkan keseluruhan subhierarki.
 | **Kueri terindeks dengan** [**pengurutan dan pemfilteran**](https://firebase.google.com/docs/firestore/client/query-data?hl=id) **gabungan.** * Anda dapat merangkai filter dan menggabungkan pemfilteran serta pengurutan pada properti dalam satu kueri.
* Menuliskan kueri dangkal untuk subkoleksi: Anda dapat membuat kueri subkoleksi dalam dokumen, dan bukan keseluruhan koleksi, atau bahkan keseluruhan dokumen.
* Kueri diindeks secara default: performa kueri sebanding dengan ukuran kumpulan hasil, bukan kumpulan data.
 |

### Operasi tulis dan transaksi

|  |  |
| --- | --- |
| **Realtime Database** | **Cloud Firestore** |
| **Operasi tulis dan transaksi dasar.** * [Penulisan data](https://firebase.google.com/docs/database/android/read-and-write?hl=id#basic_write) sebagai operasi individual.
* [Transaksi](https://firebase.google.com/docs/database/ios/read-and-write?hl=id#save_data_as_transactions) di SDK asli memerlukan callback penyelesaian.
 | **Operasi tulis dan transaksi menyeluruh.** * [Operasi batch](https://firebase.google.com/docs/firestore/manage-data/update-data?hl=id#batch_multiple_write_operations) dan penyelesaian operasi secara menyeluruh.
* [Transaksi](https://firebase.google.com/docs/firestore/manage-data/update-data?hl=id#update_data_with_transactions) mengulang secara otomatis hingga operasi selesai.
 |

### Keandalan dan performa

|  |  |
| --- | --- |
| **Realtime Database** | **Cloud Firestore** |
| **Realtime Database adalah solusi regional.** * Latensi sangat rendah, sehingga ini adalah pilihan tepat untuk sinkronisasi status yang sering dilakukan.
* Database terbatas sesuai ketersediaan menurut zona di satu region.

Baca lebih lanjut karakteristik performa dan keandalan Realtime Database dalam [Perjanjian Tingkat Layanan](https://firebase.google.com/terms/service-level-agreement?hl=id).  | **Cloud Firestore adalah solusi multi-region yang mendukung automatic scaling.** * Menyimpan data Anda di beberapa pusat data di region yang berbeda, sehingga memastikan skalabilitas global dan keandalan yang kuat.
* Menawarkan stabilitas dan keandalan Google Cloud Platform.

Baca lebih lanjut karakteristik performa dan keandalan Cloud Firestore dalam [Perjanjian Tingkat Layanan](https://cloud.google.com/firestore/sla?hl=id).  |

### Skalabilitas

|  |  |
| --- | --- |
| **Realtime Database** | **Cloud Firestore** |
| **Penskalaan membutuhkan sharding.** Skala ke sekitar 100.000 koneksi serentak dan 1.000 tulis/detik dalam database tunggal. Penskalaan yang lebih besar daripada itu memerlukan sharding data di seluruh database. | **Penskalaan berjalan otomatis.** Penskalaan sepenuhnya otomatis. Saat ini, batas penskalaan adalah sekitar 1 juta koneksi serentak dan 10.000 tulis/detik. Kami berencana meningkatkan batas ini di masa mendatang. |

### Keamanan

|  |  |
| --- | --- |
| **Realtime Database** | **Cloud Firestore** |
| **Aturan menurun yang memerlukan validasi terpisah.** * [Aturan Firebase Database](https://firebase.google.com/docs/database/security/?hl=id) adalah satu-satunya opsi keamanan.
* Aturan baca dan tulis bersifat menurun
* Anda harus [memvalidasi data](https://firebase.google.com/docs/database/security/securing-data?hl=id#validating_data) secara terpisah menggunakan aturan validate.
 | **Keamanan yang lebih sederhana dan lebih kuat untuk SDK seluler, web, dan server.** * SDK seluler dan web menggunakan [Aturan Keamanan Cloud Firestore](https://firebase.google.com/docs/firestore/security/get-started?hl=id). SDK Server menggunakan [Pengelolaan Akses dan Identitas (IAM)](https://cloud.google.com/firestore/docs/security/iam?hl=id).
* Aturan tidak menurun, kecuali Anda menggunakan karakter pengganti.
* Validasi data terjadi secara otomatis.
* Aturan dapat membatasi kueri: Jika hasil kueri mungkin berisi data yang tidak dapat diakses oleh pengguna, keseluruhan kueri akan gagal.
 |

### Harga

Harga kedua solusi tersedia di bagian [paket harga Spark, Flame, dan Blaze](https://firebase.google.com/pricing?hl=id).

|  |  |
| --- | --- |
| **Realtime Database** | **Cloud Firestore** |
| Membebankan biaya hanya untuk **bandwidth dan penyimpanan**, tetapi pada tarif lebih tinggi. Baca lebih lanjut [paket harga Realtime Database](https://firebase.google.com/pricing?hl=id). | Membebankan biaya terutama pada **operasi yang dilakukan di database Anda (baca, tulis, hapus)**, dan pada tarif, bandwidth serta penyimpanan yang lebih rendah. Cloud Firestore mendukung [batas pengeluaran](https://cloud.google.com/appengine/docs/standard/python/console/?hl=id#setting_a_spending_limit) harian untuk project Google App Engine, sehingga biayanya dipastikan tidak akan melampaui anggaran Anda.Baca lebih lanjut [paket harga Cloud Firestore](https://firebase.google.com/docs/firestore/pricing?hl=id). |

## Menggunakan Cloud Firestore dan Realtime Database

Anda dapat menggunakan kedua database ini dalam aplikasi atau project Firebase yang sama. Kedua database NoSQL ini dapat menyimpan jenis data yang sama dan library klien bekerja dengan cara yang sama. Ingatlah perbedaan yang diuraikan di atas jika Anda memutuskan untuk [menggunakan kedua database ini dalam aplikasi Anda](https://firebase.google.com/docs/firestore/firestore-for-rtdb?hl=id).

# Penginstalan & Penyiapan di Android

## Menghubungkan aplikasi ke Firebase

1. [Instal Firebase SDK](https://firebase.google.com/docs/android/setup?hl=id).
2. Pada [konsol Firebase](https://console.firebase.google.com/?hl=id), tambahkan aplikasi ke proyek Firebase.

## Menambahkan Realtime Database ke aplikasi

Tambahkan dependensi untuk Realtime Database ke file build.gradle tingkat aplikasi:

  implementation 'com.google.firebase:firebase-database:16.1.0'

## Mengonfigurasi Aturan Firebase Database

Realtime Database menyediakan bahasa aturan deklaratif, sehingga Anda dapat menentukan cara data disusun, diindeks, serta kapan data tersebut dapat dibaca dan ditulisi.

Secara default, akses baca dan tulis ke database Anda dibatasi, sehingga hanya pengguna yang terautentikasi yang dapat membaca atau menulis data. Untuk memulai tanpa menyiapkan [Authentication](https://firebase.google.com/docs/auth/?hl=id), Anda dapat [mengonfigurasi aturan untuk akses publik](https://firebase.google.com/docs/database/security/quickstart?hl=id#sample-rules%22). Tindakan ini membuat database terbuka bagi siapa saja, termasuk orang yang tidak menggunakan aplikasi Anda. Oleh karena itu, pastikan kembali untuk membatasi database Anda setelah menetapkan autentikasi.

## Menulis ke database Anda

Mengambil instance database menggunakan getInstance() dan referensi lokasi tempat Anda ingin menulis.

[Java](https://firebase.google.com/docs/database/android/start?hl=id#java-android)[Kotlin](https://firebase.google.com/docs/database/android/start?hl=id#kotlin-android)

// Write a message to the database
FirebaseDatabase database = FirebaseDatabase.getInstance();
DatabaseReference myRef = database.getReference("message");

myRef.setValue("Hello, World!");

[MainActivity.java](https://github.com/firebase/snippets-android/blob/5cbfbc0207b843602a731af031431d0026d8af94/database/app/src/main/java/com/google/firebase/referencecode/database/MainActivity.java#L40-L44)

Anda bisa menyimpan berbagai tipe data ke database dengan cara ini, termasuk objek Java. Ketika Anda menyimpan sebuah objek, respons dari setiap getter akan disimpan sebagai turunan dari lokasi ini.

## Membaca dari database Anda

Agar data aplikasi diperbarui secara real-time, Anda harus menambahkan [ValueEventListener](https://firebase.google.com/docs/reference/android/com/google/firebase/database/ValueEventListener?hl=id) ke referensi yang baru saja dibuat.

Metode onDataChange() di class ini dipicu 1 kali saat listener ditambahkan dan dipicu lagi setiap kali terjadi perubahan data, termasuk turunannya.

[Java](https://firebase.google.com/docs/database/android/start?hl=id#java-android)[Kotlin](https://firebase.google.com/docs/database/android/start?hl=id#kotlin-android)

// Read from the database
myRef.addValueEventListener(new ValueEventListener() {
    @Override
    public void onDataChange(DataSnapshot dataSnapshot) {
        // This method is called once with the initial value and again
        // whenever data at this location is updated.
        String value = dataSnapshot.getValue(String.class);
        Log.d(TAG, "Value is: " + value);
    }

    @Override
    public void onCancelled(DatabaseError error) {
        // Failed to read value
        Log.w(TAG, "Failed to read value.", error.toException());
    }
});

[MainActivity.java](https://github.com/firebase/snippets-android/blob/5cbfbc0207b843602a731af031431d0026d8af94/database/app/src/main/java/com/google/firebase/referencecode/database/MainActivity.java#L48-L63)

## Opsional: Mengonfigurasi ProGuard

Ketika menggunakan Firebase Realtime Database di aplikasi Anda bersama dengan ProGuard, Anda perlu mempertimbangkan bagaimana objek model akan dipecah dan disatukan setelah obfuscation dilakukan. Jika menggunakan <DataSnapshot.getValue(Class) atau DatabaseReference.setValue(Object) untuk membaca dan menulis data, Anda perlu menambahkan aturan ke file proguard-rules.pro:

    # Add this global rule
    -keepattributes Signature

    # This rule will properly ProGuard all the model classes in
    # the package com.yourcompany.models. Modify to fit the structure
    # of your app.
    -keepclassmembers class com.yourcompany.models.\*\* {
      \*;
    }

## Persiapan untuk Peluncuran

Sebelum meluncurkan aplikasi, sebaiknya Anda membaca [checklist peluncuran](https://firebase.google.com/support/guides/launch-checklist?hl=id) untuk memastikan aplikasi siap diluncurkan.

## Langkah Berikutnya

* Mempelajari cara [membuat struktur data](https://firebase.google.com/docs/database/android/structure-data?hl=id) untuk Realtime Database.
* [Menskalakan data Anda di beberapa instance database](https://firebase.google.com/docs/database/usage/sharding?hl=id).
* [Membaca dan menulis data](https://firebase.google.com/docs/database/android/read-and-write?hl=id).
* [Melihat database di Firebase console](https://console.firebase.google.com/project/_/database/data?hl=id).
* Jika Anda mengupgrade dari versi Firebase SDK 2.X, baca [panduan upgrade untuk Android](https://firebase.google.com/support/guides/firebase-android?hl=id).

# Membuat Struktur Database Anda

Panduan ini mencakup beberapa konsep utama dalam arsitektur data dan praktik terbaik untuk merancang struktur data JSON dalam Firebase Realtime Database Anda.

Membuat database yang terstruktur dengan baik membutuhkan perencanaan yang matang. Yang terpenting, Anda harus merencanakan bagaimana data akan disimpan dan diambil kembali untuk membuat proses tersebut menjadi semudah mungkin.

## Metode perancangan struktur data: dengan pohon JSON

Semua data Firebase Realtime Database disimpan sebagai objek JSON. Anda dapat menganggap database sebagai pohon JSON yang dihosting di cloud. Tidak seperti database SQL, database tersebut tidak memiliki tabel atau catatan. Ketika Anda menambahkan data ke pohon JSON, data tersebut akan menjadi node di struktur JSON yang ada dengan kunci terkait. Anda dapat memasukkan kunci sendiri, misalnya ID pengguna atau nama semantik, atau kunci tersebut dapat tersedia untuk Anda menggunakan [push()](https://firebase.google.com/docs/reference/android/com/google/firebase/database/DatabaseReference.html?hl=id#push()).

# Membaca dan Menulis Data di Android

Dokumen ini membahas dasar-dasar dalam membaca dan menulis data Firebase.

Data Firebase ditulis ke referensi FirebaseDatabase dan diambil dengan memasang listener asinkron ke referensi. Listener terpicu sekali untuk status awal data dan terpicu lagi setiap kali data berubah.