Firebase Database

Penerapan model Basis Data NoSQL diwujudkan dengan penyederhanaan desain yang digunakan. Desain database NoSQL tidak seperti yang diterapkan pada RDBMS. Penyimpanan data dilakukan dalam bentuk yang tidak terstruktur, dan sering dilakukan bentuk denormalisasi dalam pemodelannya. Sebuah model dari data yang disimpan dalam RDBMS, dibentuk berdasarkan struktur datanya dan diterapkan normalisasi untuk menghilangkan redundansi/duplikasi data. Akan tetapi, model dari NoSQL memiliki pendekatan yang berbeda. Struktur skema biasanya dibentuk berdasarkan “bagaimana cara menampilkan data”. Sehingga, secara tidak langsung, terlihat bahwa cara menyimpan data memperhitungkan cara simple untuk mengambil data yang akan ditampilkan.

Firebase merupakan salah satu bentuk dari NoSQL (diasosiasikan dengan Not Only SQL). Penyimpanan data pada Firebase dilakukan dalam format JSON (JavaScript Object Notation). JSON merupakan lightweight interchange data format. Dengan kata lain, JSON merupakan format pertukaran data yang cukup “ringan” dalam penggunaannya. Pada modul ini, dijelaskan beberapa aspek dari penggunaan Firebase database, dan demo pengaksesannya menggunakan Developer Tools dari Web Browser Google Chrome.

# A. Pengenalan syntax JSON

JSON merupakan format pertukaran data, yang menyimpan data dalam bentuk pasangan key, dan value. Key dan value dipisahkan oleh tanda “:”. Value memiliki tipe data yang dituliskan sebagai berikut:

1. Tipe data Objek: ditandai dengan diapit oleh { dan }
2. Tipe data Array: ditandai dengan diapit oleh [ dan ]
3. Tipe data Primitif
   1. Tipe data string: ditandai diapit oleh “ dan “
   2. Tipe data numerik: berupa angka tanpa diapit symbol apapun
   3. Tipe data Boolean: berupa value true dan false

Berdasarkan keterangan di atas, terlihat bahwa JSON, secara khusus tidak memiliki tipe data “DATE”. Untuk mengatasi hal ini, penyimpanan tanggal pada JSON dapat dilakukan dengan beberapa cara. Cara yang paling sering dilakukan yaitu (1) penyimpanan date dalam bentuk string literal dari tanggal dengan format tertentu, atau (2) menyimpan dalam bentuk long untuk menyimpan nilai unix time dari tanggal tersebut.

Contoh dari tipe data JSON adalah sebagai berikut:

|  |
| --- |
| {      **"name"**:"John",    **"age"**:31,    **"city"**:"New York" } |

Tipe data JSON di atas menyimpan 1 buah objek (ditandai dengan kurung kerawal/curly bracket), dengan key berupa “name”, “age”, dan “city”. Masing-masing key, memiliki value yang berurutan yaitu: “John”, 31, dan juga “New York”. Kode JSON dapat diterapkan secara bertingkat. Dengan kata lain, sebuah Objek, dapat memiliki value berupa objek lain. Syntax JSON di atas diterapkan pada penyimpanan data Firebase. Hal yang perlu diperhatikan yaitu, setiap key memiliki nama yang berbeda untuk 1 tingkat node yang sama (tingkat sibling).

Contoh:

|  |
| --- |
| {      **"users"**:{         **"alovelace"**:{            **"name"**:"Ada Lovelace",          **"contacts"**:{               **"ghopper"**:true          }       }    } } |

# B. Memodelkan penyimpanan data

Firebase memetakan pemodelan dalam bentuk kode JSON. Berbeda dengan penyimpanan RDBMS yang membutuhkan DDL, dan baru melakukan penyimpanan data menggunakan DML. Sebagai bentuk dari NoSQL, Firebase sering disebut sebagai schemaless. Artinya, tidak digunakan query DDL untuk menyimpan data. Walaupun struktur & skema tetap dibutuhkan untuk dimodekan agar penyimpanan dan pengambilan data simple untuk dilakukan. Secara umum, tidak ada aturan khusus bagaimana cara menyimpan suatu data pada Firebase. Aturan khusus yang dimaksud adalah ketentuan 1NF, 2NF, penentuan functional dependency, dan lainnya.

Akan tetapi, sebagai SQL Deveoper pemula, terdapat beberapa analogi yang memudahkan proses transisi untuk memahami cara penyimpanan dari Firebase ini. Contoh, terdapat sebuah tabel dengan structure (struktur tabel) dan record (isi tabel) sebagai berikut:

|  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| Struktur Tabel   |  |  | | --- | --- | | **Users** | | | ID | Int | | firstname | Varchar(100) | | lastname | Varchar(100) | | age | Int | | Isi Tabel   |  |  |  |  | | --- | --- | --- | --- | | **ID** | **firstname** | **lastname** | **age** | | 1 | Naruto | Uzumaki | 18 | | 2 | Hatake | Kakashi | 25 | | 3 | Haruno | Sakura | 17 | |

Model tabel di atas, dapat dituliskan sebagai berikut pada Firebase dengan menggunakan notasi JSON:

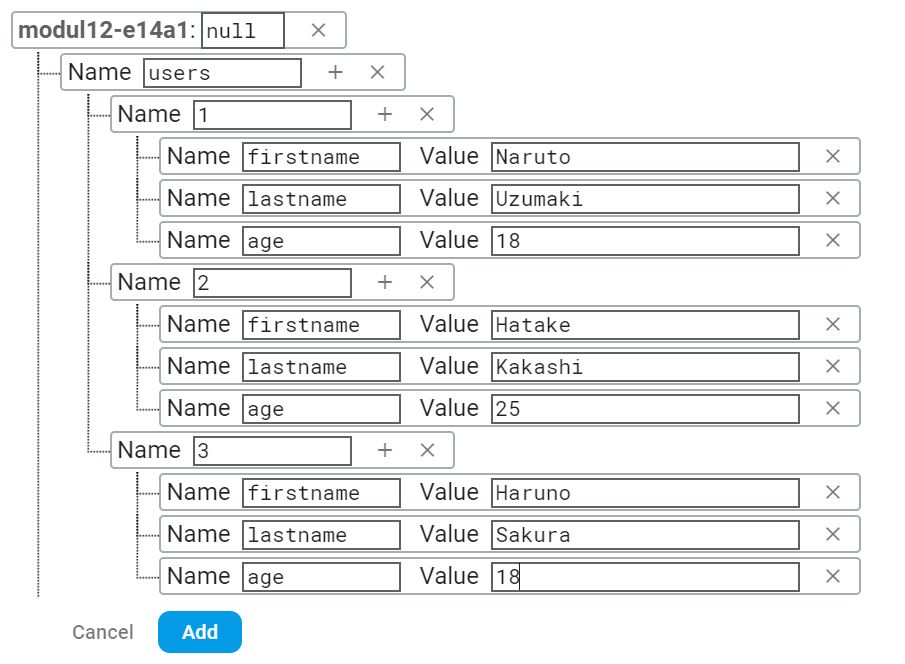
|  |
| --- |
| {  "users" : {  "1" : {  "age" : 18,  "firstname" : "Naruto",  "lastname" : "Uzumaki"  },  "2" : {  "age" : 25,  "firstname" : "Hatake",  "lastname" : "Kakashi"  },  "3" : {  "age" : 18,  "firstname" : "Haruno",  "lastname" : "Sakura"  }  } |

Membuat data pada Firebase (GUI) dengan menggunakan :

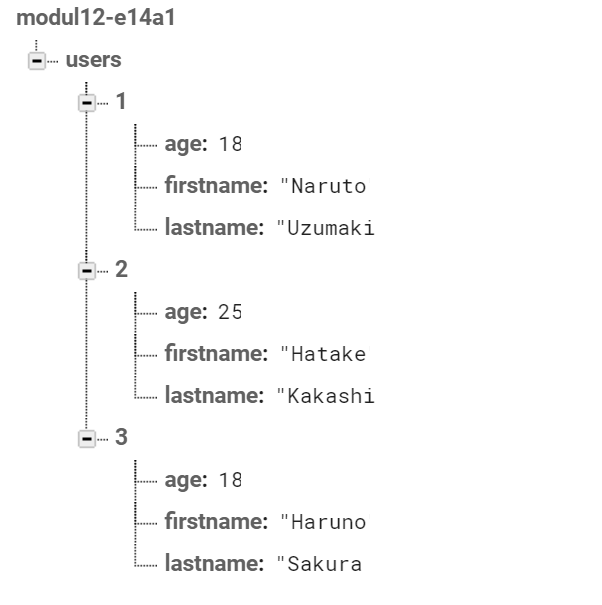
Langkah 1: Akses [https://console.firebase.google.com](https://console.firebase.google.com/u/1/). Buat project di Firebase. Berikan nama project yang bersesuaian:

Langkah 2: Pada menu navigasi “Develop” di sebelah kiri layar, pilih “Database”. Atau, juga dapat memilih “Cloud Firestore” pada menu utama. Pilih “Create Database”. Atur security database. Untuk kali ini, pilih “Test” mode. **Note**: data yang dapat digunakan di sini bisa berupa realtime database (firebase’s original database) atau cloud firestore (firebase’s new flagship database). Pada modul ini dapat dipilih 2 mode, hanya saja di bagian tertentu, fokus pada pembahasan realtime database.

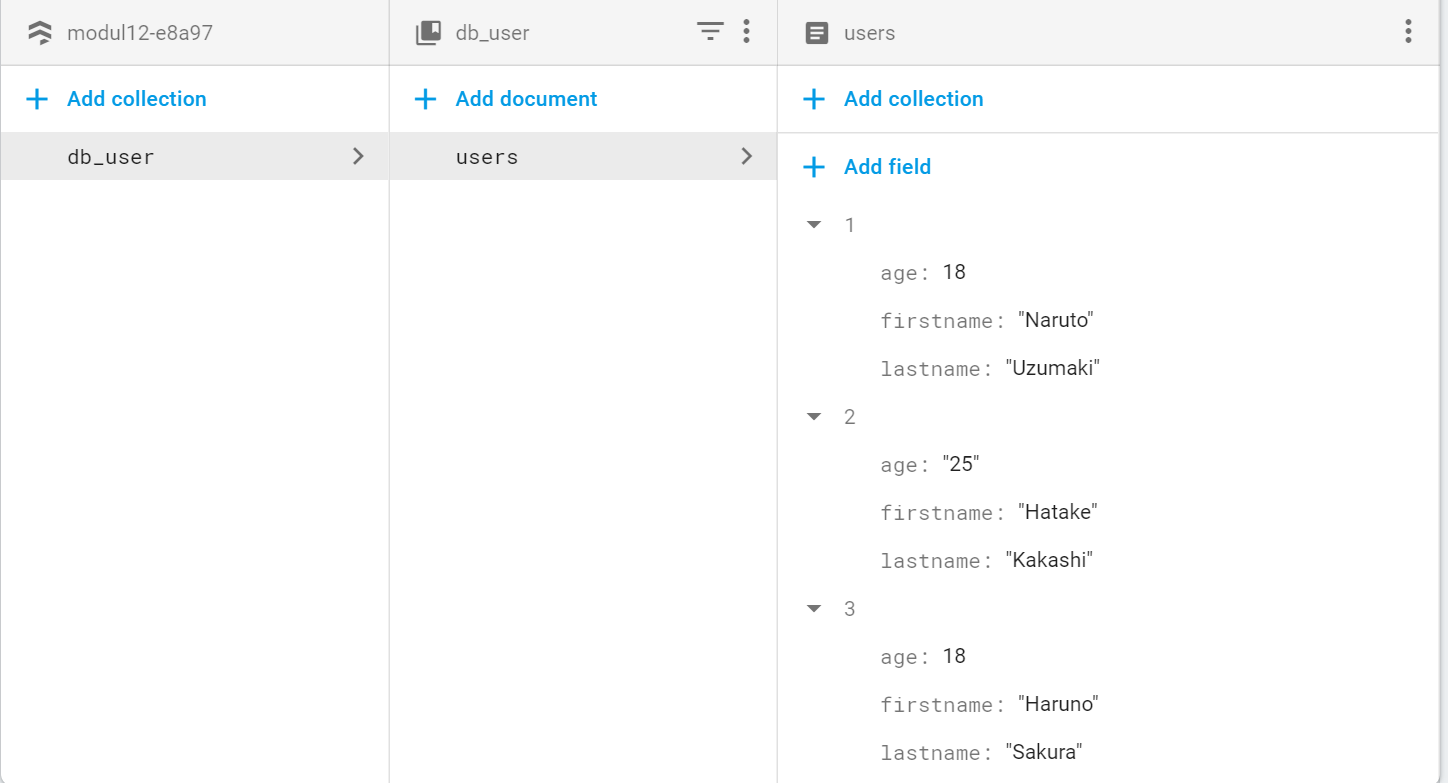
Langkah 3 (realtime database): Buat struktur sebagai berikut, lalu tekan “Add”:



Biasanya, dianalogikan child pertama dari modul adalah nama tabel yang digunakan. Dalam hal ini adalah “users”. Child berikutnya berisi PRIMARY KEY dari tabel yang digunakan. Dan berikutnya, berisi nilai record dari setiap baris. Hasil dari bentuk node di atas adalah sebuah JSON sebagai berikut:

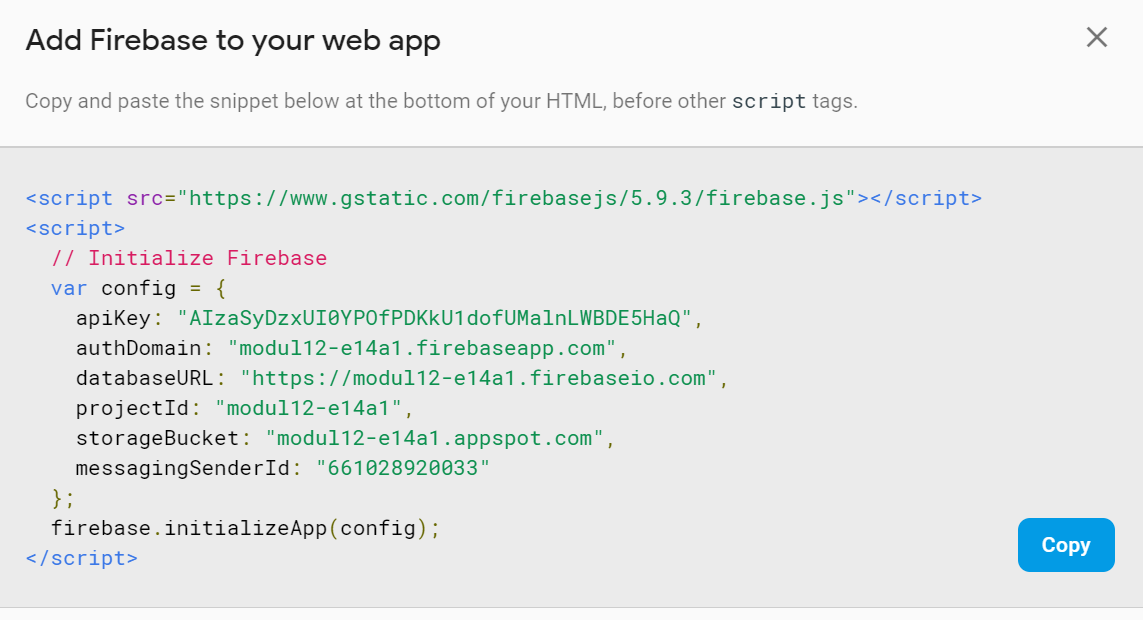


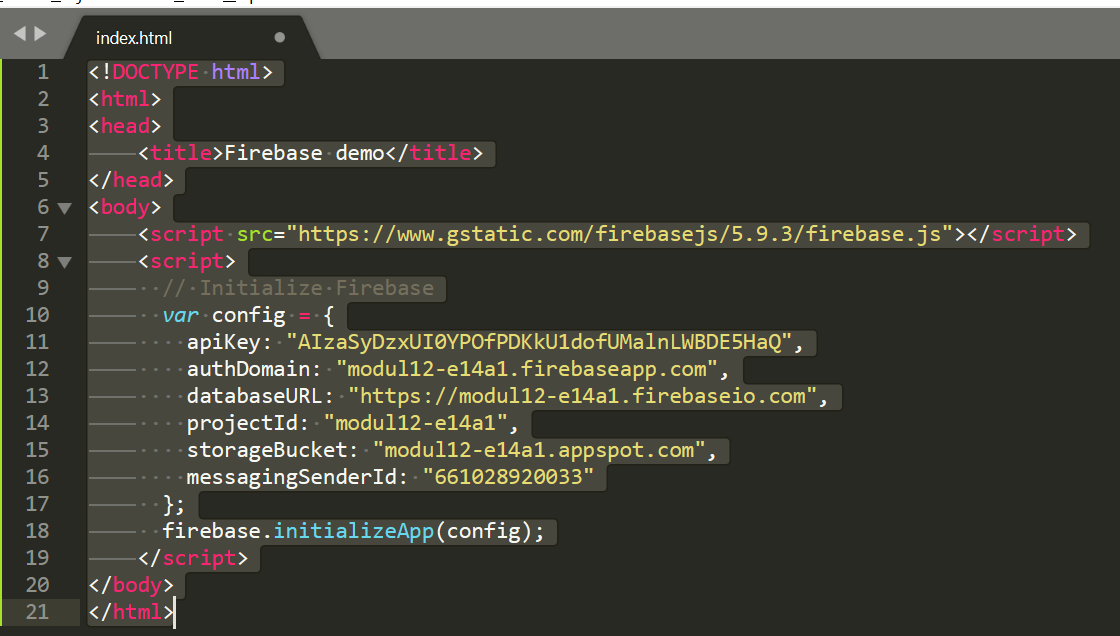
Langkah 3 (cloud Firestore): Pilih Add Collection. Masukkan “users” pada collection id. Bagian ini, akan membuat nama “database” atau “schema” dari versi RDBMS. Pada document id, berikan “users”. Pada bagian document id, ditentukan “nama tabel” dari versi RDBMS. Pada bagian field, berikan nilai “1” dengan tipe “map”. Pada bagian ini, 1 dianalogikan sebagai PRIMARY KEY, dan pada node berikutnya, tuliskan semua nilai kolom dan record yang disimpan.



Note: terdapat pendekatan lain terhadap analogi ini. Ada yang menjadikan dokumen sebagai referensi terhadap PK, dan collection sebagai 1 record. Hal ini berbeda tergantung kesepakatan dari developer aplikasi.

Data di atas dapat digunakan untuk aplikasi web/desktop/mobile. Pada modul ini digunakan pengambilan dengan menggunakan aplikasi web (html file). Pertama, persiapkan file untuk mengambil data, lalu copy-paste bagian untuk initialize firebase ke kode tersebut. Kode berikut, didapat dari halaman awal project dari “get started”. Konfigurasi ini berbeda untuk setiap project.



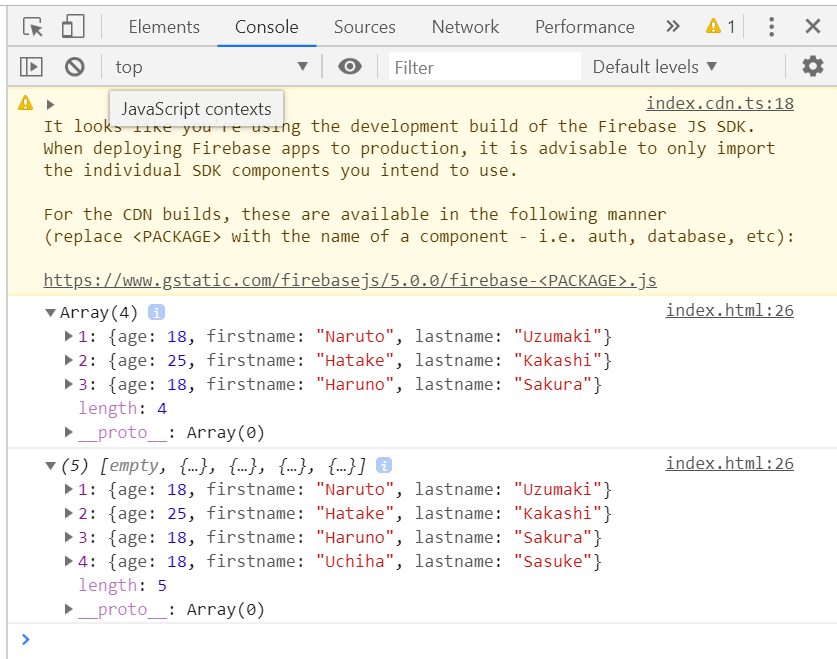


Tambahkan kode untuk melakukan pengaksesan, sebelum tag penutup </script>:

|  |
| --- |
| ……  //reference database  var db = firebase.database();  var usersRef = db.ref('users');  usersRef.on('value',showData, showError);  function showData(items){  console.log(items.val())  }  function showError(err){  console.log(err);  }  ……. |

Hasilnya dapat dilihat setelah menjalankan file web, dan dikarenakan ditampilkan dalam console, maka cek developer tools pada tab console. Maka terlihat data yang sudah tersimpan. Data ini dapat digunakan untuk sebuah aplikasi, baik itu web ataupun mobile.

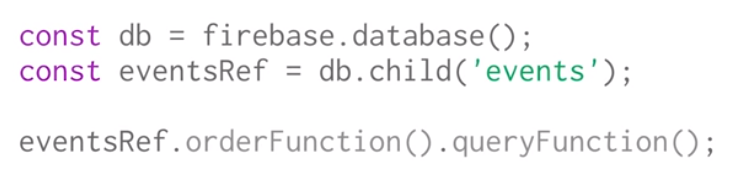
Hal unik dari firebase adalah sifatnya yang realtime. Jika terdapat perubahan dari data yang diberikan, maka pada aplikasi yang menggunakan data tersebut akan langsung terdapat perubahan dari data yang ditampilkan. Contoh, tambahkan 1 data pada firebase. Selama masih terkoneksi ke internet, maka saat pergi ke browser, langsung terdapat perubahan data



# C. Melakukan query

Pada penyimpanan, hal yang diperlukan adalah melakukan query. Berdasarkan struktur tabel di atas, pada modul ini akan dituliskan beberapa contoh query yang mungkin untuk dilakukan.

Pada RDBMS, secara umum, digunakan 2 tahapan dalam query suatu record, yaitu melakukan pemilihan kolom pada tabel (SELECT x FROM y), lalu memberikan batasan dari record yang akan diambil (WHERE z). Begitupula dengan Firebase, tahapan terdiri dari 2, yaitu pemberian reference bagi parent key, lalu gunakan ordering function. Selain itu, juga dapat ditambahkan querying function pada kode pengambilan data:



Terdapat 4 ordering function yang dapat digunakan: orderByKey(), orderByChild(‘childProperty’), orderByValue, dan orderByPriority. Akan tetapi, fungsi terakhir merupakan cara lama dan dapat digantikan dengan 3 fungsi pertama. Sedangkan untuk querying function yang dapat digunakan adalah: startAt(‘value’), endAt(‘value’), equalTo(‘child\_key’), limitToFirst(number), limitToLast(number). Query RDBMS dapat dianalogikan pada query Firebase. Contohnya, kedua query berikut bersifat mengambil data yang sama:

|  |  |
| --- | --- |
|  |  |

Pada modul ini akan dibahas beberapa contoh dengan menggunakan data yang sudah dibuat sebelumnya di atas.

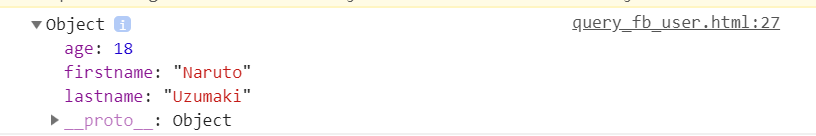
**Query 1: Mengambil keseluruhan record dengan atribut PRIMARY KEY.**

|  |
| --- |
| Query RDBMS:  SELECT \* FROM users WHERE **id=1;**  Query (dituliskan pada script JS/TS di file web):  var db = firebase.database();  var usersRef = db.ref('users').**child('1')**; |

Penggunaan:



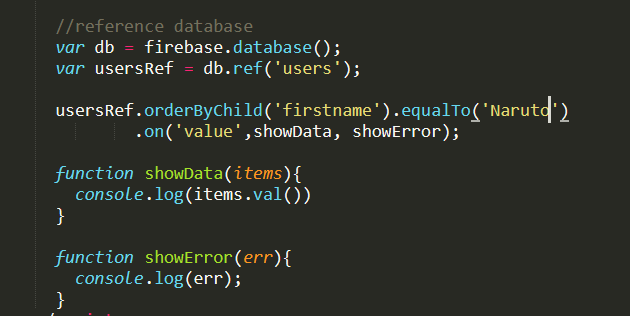
Hasil:



**Query 2: Mengambil keseluruhan record dengan parameter tertentu.**

|  |
| --- |
| Query RDBMS:  SELECT \* FROM users WHERE **firstname=’Naruto’;**  Query (dituliskan pada script JS/TS di file web):  var db = firebase.database();  var usersRef = db.ref('users').**orderByChild(‘firstname’).equalTo(‘Naruto’)**; |

Penggunaan:



Hasil:



Pada hasil di atas, terdapat warning. Untuk menghilangkan warning tersebut, dapat ditambahkan rule di Firebasenya:

|  |
| --- |
| {  "rules": {  ".read": true,  ".write": true,  "users":{  ".indexOn": "firstname"  }  }  } |

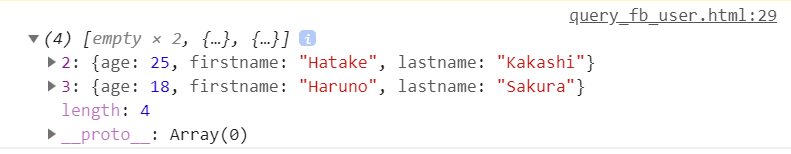
**Query 3: Mengambil keseluruhan record dengan parameter tertentu menggunakan kondisi LIKE.**

|  |
| --- |
| Query RDBMS:  SELECT \* FROM users WHERE **firstname LIKE ‘H%’;**  Query (dituliskan pada script JS/TS di file web):  var db = firebase.database();  usersRef.**orderByChild('firstname').startAt('H').endAt('H\uf8ff')**  .on('value',showData, showError); |

Penggunaan:



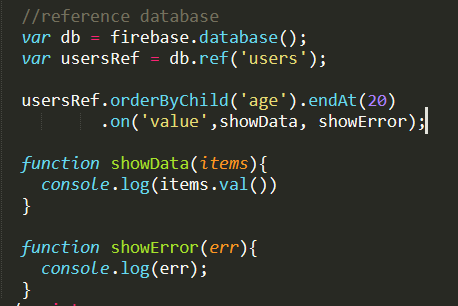
Hasil:



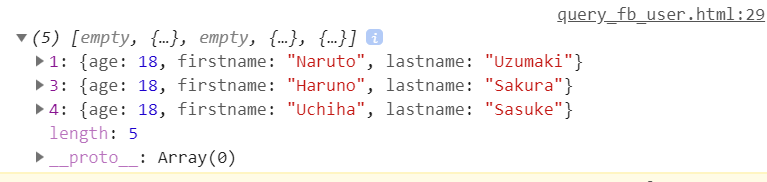
**Query 4: Mengambil keseluruhan record dengan pengkondisian angka.**

|  |
| --- |
| Query RDBMS:  SELECT \* FROM users WHERE **age <20;**  Query (dituliskan pada script JS/TS di file web):  var db = firebase.database();  usersRef.**orderByChild('age').endAt(20)**  .on('value',showData, showError); |

Penggunaan:



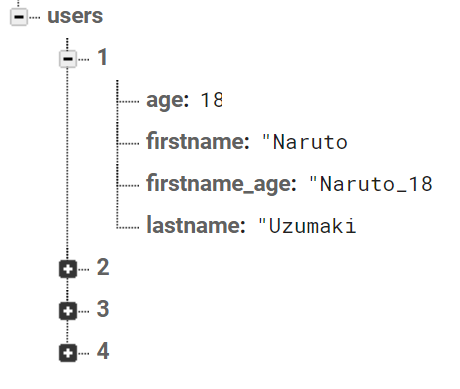
Hasil:



**Query 5: Menggunakan Operator Pengkondisian**

|  |
| --- |
| Query RDBMS:  SELECT \* FROM users WHERE **firstname = ‘Naruto’ AND** **age < 20;**  Query (dituliskan pada script JS/TS di file web): Tidak ada yang bersesuaian |

Jika harus diusahaan, untuk menggunakan operasi AND di sini, struktur pada data Firebase harus diubah. Modifikasi dilakukan dengan mengubah struktur sebagai berikut untuk kondisi di atas:

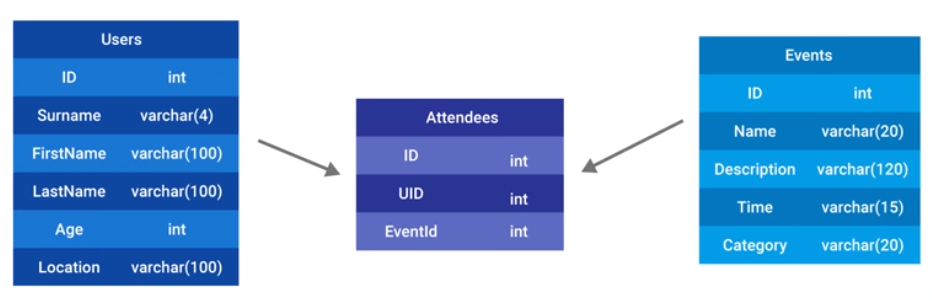
 sehingga query dapat diberikan:

|  |
| --- |
| //reference database  var db = firebase.database();  var usersRef = db.ref('users');  usersRef.orderByChild('firstname\_age').equalTo('Naruto\_18')  .on('value',showData, showError); |

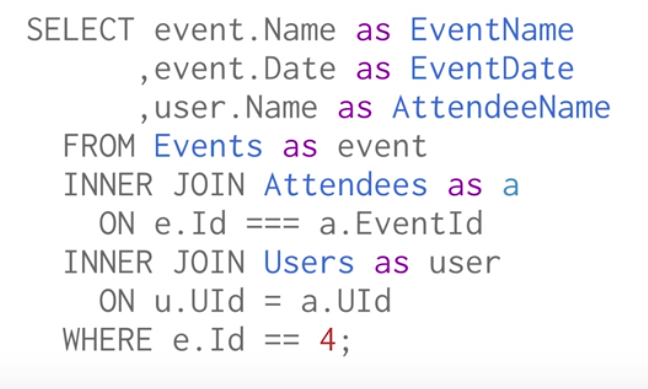
Hal di atas memperlihatkan bahwa aturan yang digunakan di Firebase memang dirujuk dari cara mengambil/menampilkan data. Bukan berdasarkan aturan normalisasi seperti yang dilakukan pada Firebase.

# D. Melakukan operasi JOIN

Untuk melakukan penyimpanan dalam 2 tabel, dengan contoh gambar skema relasi sebagai berikut:



Dengan query JOIN yang dapat dituliskan sebagai berikut:

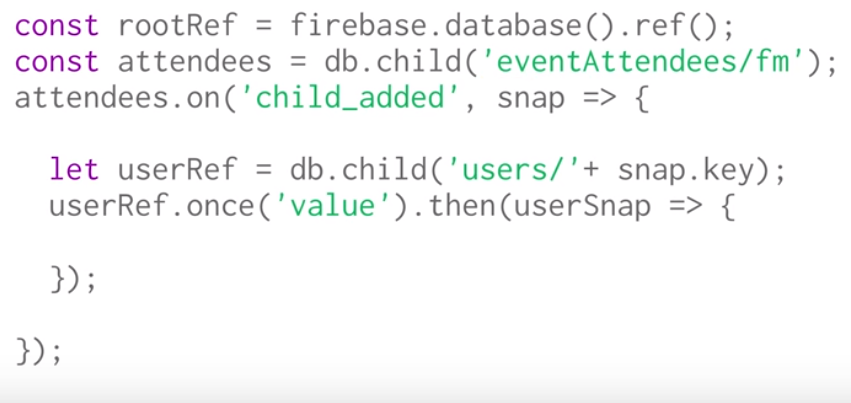


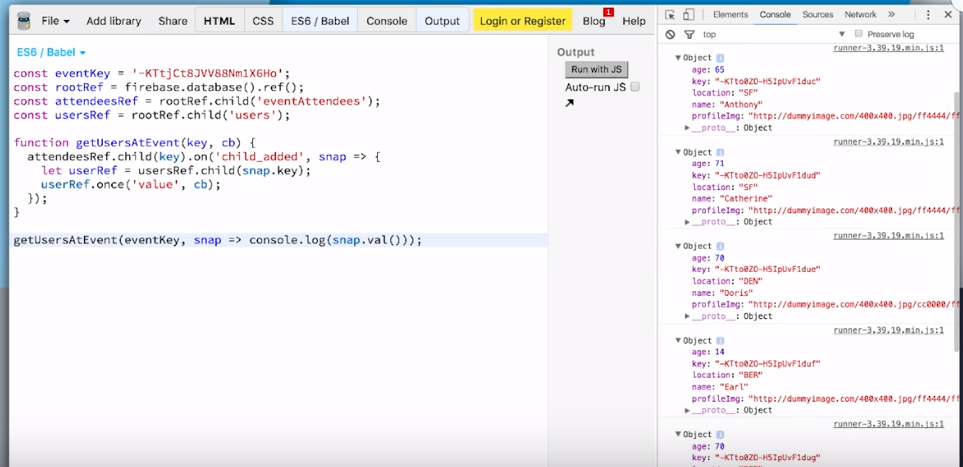
Penyimpanan dalam bentuk Firebase dapat dilakukan sebagai berikut:



Pada contoh di atas, terlihat bahwa pada “eventAttendees”, terdapat penyimpanan key event dari node “events”, berupa key “fm”. Dan masih pada “eventAttendees”, terihat ada key “users” sebagai child dari node “fm” tersebut. Hal ini untuk memperlihatkan event “Firebase Meetup” didatangi oleh “David” dan “Alice” merujuk dari value node sebelumnya.

Untuk mengambil data user yang menghadiri event tertentu, dapat digunakan kode berikut:





Selain menggunakan cara di atas, dapat juga dilakukan dengan melakukan bentuk penyimpanan bertingkat sebagai berikut:

|  |  |
| --- | --- |
|  |  |

Pada contoh di atas, hal yang dilakukan adalah menyimpan objek user secara keseluruhan. Tidak lagi hanya menggunakan penyimpanan berupa id, akan tetapi juga nilai objek yang dibutuhkan. Hal ini disebut normalisasi. Bandingkan penyimpanan di sebelah kiri (struktur awal), dengan penyimpanan di sebelah kanan (hasil denormalisasi). Denormalisasi dilakukan setelah perancangan tampilan dilakukan, sesuai kebutuhan dari pengguna akhir. Hal ini akan mempengaruhi bagaimana cara pengambilan data. Tidak ada aturan khusus bagaimana sebuah data disimpan. Terdapat beberapa saran bagaimana data disimpan dalam bentuk JOIN, tapi tidak ada ketentuan pasti untuk hal ini. Efek dari perubahan struktur ini, programmer harus melakukan update multi path terhadap data terkait.

Referensi:

1. <https://www.w3schools.com/js/js_json_intro.asp>
2. <https://firebase.google.com/docs/database/rest/start?hl=id>
3. Playlist “Kenalan dengan Firebase” pada channel Youtube “Sekolah Koding”
4. Playlist “The Firebase Database For SQL Developers” pada channel Youtube “Firebase”

# Latihan/LAB’S WORK

**No 1** Diketahui struktur tabel sebagai berikut:

|  |
| --- |
| PAMERAN  nama: varchar2 50 karakter  tanggal: date  nomor: char 5 karakter |

Dengan data yang disimpan sebagai berikut:

|  |  |  |
| --- | --- | --- |
| NOMOR | NAMA | TANGGAL |
| PMR01 | Seni Berubah. Kamu Juga | 2018-10-09 |
| PMR02 | Indonesia Contemporary Art | 2017-01-03 |
| PMR03 | Youth’s Perspective | 2018-09-09 |
| PMR04 | Biennale Bandung 2017 | 2017-02-21 |
| PMR05 | Biennale Bandung 2019 | 2019-04-18 |

1. Berdasarkan informasi di atas, tuliskan bentuk struktur data yang dapat disimpan dalam Firebase-nya.
2. Tuliskan query untuk menghasilkan query berikut:
   1. Pameran yang dilakukan di tahun 2018
   2. Pameran yang Namanya diawali huruf ‘B’
   3. Data nama dan tanggal pameran dengan nomor ‘PMR03’
   4. Tampilkan semua data pameran yang ada

**No 2** Jika diketahui skema relasi tabel berikut (hubungan n-m), sarankanlah bagaimana bentuk penyimpanan data di Firebase. Jelaskan pula alasan mengapa menggunakan struktur data tersebut.

