Consulta de dados

Com consultas no banco de dados do Firebase, podemos recuperar dados seletivamente de acordo com vários fatores. Para criar uma consulta no seu banco de dados, comece especificando a ordenação dos dados usando uma das funções de ordenação: orderByChild(), orderByKey() ou orderByValue(). Em seguida, combine-as com outros cinco métodos para realizar consultas complexas: limitToFirst(), limitToLast(), startAt(), endAt() e equalTo().

Já que a equipe do Firebase gosta muito de dinossauros, usaremos um snippet de uma amostra de banco de dados de fatos sobre esses animais para demonstrar como consultar dados no seu banco de dados do Firebase.

{  
  "lambeosaurus": {  
    "height" : 2.1,  
    "length" : 12.5,  
    "weight": 5000  
  },  
  "stegosaurus": {  
    "height" : 4,  
    "length" : 9,  
    "weight" : 2500  
  }  
}

Você pode ordenar os dados de três formas: por **chave filha**, por **chave** ou por **valor**. Uma consulta básica do banco de dados começa com uma dessas funções de ordenação. Todas elas são explicadas a seguir.

Ordenação por uma chave filha especificada

Você pode ordenar os nós por uma chave filha comum, transmitindo essa chave para orderByChild(). Por exemplo, para ler todos os dinossauros ordenados por altura, faça o seguinte:

[Java](https://firebase.google.com/docs/database/admin/retrieve-data?hl=pt-br#java-java)[Node.js](https://firebase.google.com/docs/database/admin/retrieve-data?hl=pt-br#node.js-node.js)[Python](https://firebase.google.com/docs/database/admin/retrieve-data?hl=pt-br#python-python)[Go](https://firebase.google.com/docs/database/admin/retrieve-data?hl=pt-br#go-go)

public static class Dinosaur {  
  
  public int height;  
  public int weight;  
  
  public Dinosaur(int height, int weight) {  
    // ...  
  }  
  
}  
final DatabaseReference dinosaursRef = database.getReference("dinosaurs");  
dinosaursRef.orderByChild("height").addChildEventListener(new ChildEventListener() {  
  @Override  
  public void onChildAdded(DataSnapshot dataSnapshot, String prevChildKey) {  
    Dinosaur dinosaur = dataSnapshot.getValue(Dinosaur.class);  
    System.out.println(dataSnapshot.getKey() + " was " + dinosaur.height + " meters tall.");  
  }  
  
  // ...  
});

Qualquer nó que não tiver a chave filha que estamos consultando será classificado com um valor null, o que significa que ele será o primeiro da ordenação. Para ver mais detalhes sobre como os dados são ordenados, consulte a seção [Ordenação dos dados](https://firebase.google.com/docs/database/admin/retrieve-data?hl=pt-br#section-ordered-data).

As consultas também podem ser ordenadas por filhos aninhados em vários níveis, em vez de apenas filhos de um nível inferior. Isso é útil se você tiver dados aninhados em vários níveis, como estes:

{  
  "lambeosaurus": {  
    "dimensions": {  
      "height" : 2.1,  
      "length" : 12.5,  
      "weight": 5000  
    }  
  },  
  "stegosaurus": {  
    "dimensions": {  
      "height" : 4,  
      "length" : 9,  
      "weight" : 2500  
    }  
  }  
}

Agora, para consultar a altura, use o caminho completo para o objeto em vez de uma única chave:

[Java](https://firebase.google.com/docs/database/admin/retrieve-data?hl=pt-br#java-java)[Node.js](https://firebase.google.com/docs/database/admin/retrieve-data?hl=pt-br#node.js-node.js)[Python](https://firebase.google.com/docs/database/admin/retrieve-data?hl=pt-br#python-python)[Go](https://firebase.google.com/docs/database/admin/retrieve-data?hl=pt-br#go-go)

dinosaursRef.orderByChild("dimensions/height").addChildEventListener(new ChildEventListener() {  
  @Override  
  public void onChildAdded(DataSnapshot dataSnapshot, String prevChildKey) {  
    // ...  
  }  
  
  // ...  
});

As consultas podem ser ordenadas apenas de chave em chave. Chamar orderByChild() várias vezes na mesma consulta causará um erro.

Uso de índices para melhorar o desempenho

Se você quiser usar **orderByChild()** em um app de produção, defina as chaves que serão indexadas por meio da regra **.indexOn** nas suas regras de segurança e do Firebase. Você pode criar essas consultas ad hoc no cliente, mas o desempenho será muito melhor se usar **.indexOn**. [Leia a documentação](https://firebase.google.com/docs/database/security/indexing-data?hl=pt-br) para saber mais informações sobre a regra **.indexOn**.

Ordenação por chave

Você também pode ordenar os nós pelas chaves usando o método orderByKey(). O exemplo a seguir lê todos os dinossauros em ordem alfabética:

[Java](https://firebase.google.com/docs/database/admin/retrieve-data?hl=pt-br#java-java)[Node.js](https://firebase.google.com/docs/database/admin/retrieve-data?hl=pt-br#node.js-node.js)[Python](https://firebase.google.com/docs/database/admin/retrieve-data?hl=pt-br#python-python)[Go](https://firebase.google.com/docs/database/admin/retrieve-data?hl=pt-br#go-go)

dinosaursRef.orderByKey().addChildEventListener(new ChildEventListener() {  
  @Override  
  public void onChildAdded(DataSnapshot dataSnapshot, String prevChildKey) {  
    System.out.println(dataSnapshot.getKey());  
  }  
  
  // ...  
});

Ordenação por valor

Você pode ordenar os nós pelo valor das chaves filhas usando o método orderByValue(). Vamos supor que os dinossauros estejam participando de uma olimpíada jurássica e que estamos monitorando as pontuações deles no seguinte formato:

{  
  "scores": {  
    "bruhathkayosaurus" : 55,  
    "lambeosaurus" : 21,  
    "linhenykus" : 80,  
    "pterodactyl" : 93,  
    "stegosaurus" : 5,  
    "triceratops" : 22  
  }  
}

Para ordenar os dinossauros por pontuação, podemos criar a seguinte consulta:

[Java](https://firebase.google.com/docs/database/admin/retrieve-data?hl=pt-br#java-java)[Node.js](https://firebase.google.com/docs/database/admin/retrieve-data?hl=pt-br#node.js-node.js)[Python](https://firebase.google.com/docs/database/admin/retrieve-data?hl=pt-br#python-python)[Go](https://firebase.google.com/docs/database/admin/retrieve-data?hl=pt-br#go-go)

DatabaseReference scoresRef = database.getReference("scores");  
scoresRef.orderByValue().addChildEventListener(new ChildEventListener() {  
  @Override  
  public void onChildAdded(DataSnapshot dataSnapshot, String prevChildKey) {  
    System.out.println("The " + dataSnapshot.getKey() + " score is " + dataSnapshot.getValue());  
  }  
  
  // ...  
});

Consulte a seção [Ordenação dos dados](https://firebase.google.com/docs/database/admin/retrieve-data?hl=pt-br#section-ordered-data) para ver uma explicação de como os valores null, booleanos, de string e de objeto são ordenados usando orderByValue().

Indexação de valores para melhorar o desempenho

Se você quiser usar **orderByValue()** em um app de produção, precisará adicionar **.value** às suas regras, no índice adequado. [Leia a documentação](https://firebase.google.com/docs/database/security/indexing-data?hl=pt-br) para saber mais informações sobre a regra **.indexOn**.

Consultas complexas

Agora que está claro como os dados são ordenados, use os métodos de **limite** ou **intervalo** descritos abaixo para criar consultas mais complexas.

Consultas de limite

As consultas limitToFirst() e limitToLast() são usadas para configurar o número máximo de filhos que serão sincronizados para um retorno de chamada específico. Se você configurar um limite de 100, inicialmente, receberá até 100 eventos child\_added somente. Se tiver menos de 100 mensagens armazenadas no banco de dados, um evento child\_added será disparado para cada mensagem. No entanto, se tiver mais de 100 mensagens, você receberá um evento child\_added somente para 100 delas. Essas mensagens serão as 100 primeiras ordenadas, se limitToFirst() for usada, ou as 100 últimas, se limitToLast() for usada. À medida que os itens mudarem, você receberá eventos child\_added para os itens que entrarem na consulta e eventos child\_removed para os itens que saírem dela. Assim, o número total continuará sendo 100.

Usando o banco de dados de fatos de dinossauro e orderByChild(), você pode encontrar os dois dinossauros mais pesados:

[Java](https://firebase.google.com/docs/database/admin/retrieve-data?hl=pt-br#java-java)[Node.js](https://firebase.google.com/docs/database/admin/retrieve-data?hl=pt-br#node.js-node.js)[Python](https://firebase.google.com/docs/database/admin/retrieve-data?hl=pt-br#python-python)[Go](https://firebase.google.com/docs/database/admin/retrieve-data?hl=pt-br#go-go)

dinosaursRef.orderByChild("weight").limitToLast(2).addChildEventListener(new ChildEventListener() {  
  @Override  
  public void onChildAdded(DataSnapshot dataSnapshot, String prevChildKey) {  
    System.out.println(dataSnapshot.getKey());  
  }  
  
  // ...  
});

O retorno de chamada child\_added será acionado exatamente duas vezes, a menos que haja menos de dois dinossauros armazenados no banco de dados. Ele também será acionado para todo dinossauro novo, com peso maior, adicionado ao banco de dados. Em Python, a consulta retorna diretamente um OrderedDict contendo os dois dinossauros mais pesados.

Do mesmo modo, você pode encontrar os dois dinossauros mais baixos usando limitToFirst():

[Java](https://firebase.google.com/docs/database/admin/retrieve-data?hl=pt-br#java-java)[Node.js](https://firebase.google.com/docs/database/admin/retrieve-data?hl=pt-br#node.js-node.js)[Python](https://firebase.google.com/docs/database/admin/retrieve-data?hl=pt-br#python-python)[Go](https://firebase.google.com/docs/database/admin/retrieve-data?hl=pt-br#go-go)

dinosaursRef.orderByChild("weight").limitToFirst(2).addChildEventListener(new ChildEventListener() {  
  @Override  
  public void onChildAdded(DataSnapshot dataSnapshot, String prevChildKey) {  
    System.out.println(dataSnapshot.getKey());  
  }  
  
  // ...  
});

O retorno de chamada child\_added é disparado exatamente duas vezes, a menos que haja menos de dois dinossauros armazenados no banco de dados. Ele também será acionado novamente se um dos dois primeiros dinossauros for removido do banco de dados, já que um novo dinossauro será o segundo mais baixo. Em Python, a consulta retorna diretamente um OrderedDict contendo os dinossauros mais baixos.

Também é possível realizar consultas limitadas com orderByValue(). Se quiser criar um placar com os três dinossauros com a maior pontuação, faça o seguinte:

[Java](https://firebase.google.com/docs/database/admin/retrieve-data?hl=pt-br#java-java)[Node.js](https://firebase.google.com/docs/database/admin/retrieve-data?hl=pt-br#node.js-node.js)[Python](https://firebase.google.com/docs/database/admin/retrieve-data?hl=pt-br#python-python)[Go](https://firebase.google.com/docs/database/admin/retrieve-data?hl=pt-br#go-go)

scoresRef.orderByValue().limitToFirst(3).addChildEventListener(new ChildEventListener() {  
  @Override  
  public void onChildAdded(DataSnapshot dataSnapshot, String prevChildKey) {  
    System.out.println("The " + dataSnapshot.getKey() + " score is " + dataSnapshot.getValue());  
  }  
  
  // ...  
});

Consultas de intervalo

O uso de startAt(), endAt() e equalTo() permite escolher pontos arbitrários de início e término para suas consultas. Por exemplo, se você quisesse encontrar todos os dinossauros com pelo menos três metros de altura, poderia combinar orderByChild() e startAt():

[Java](https://firebase.google.com/docs/database/admin/retrieve-data?hl=pt-br#java-java)[Node.js](https://firebase.google.com/docs/database/admin/retrieve-data?hl=pt-br#node.js-node.js)[Python](https://firebase.google.com/docs/database/admin/retrieve-data?hl=pt-br#python-python)[Go](https://firebase.google.com/docs/database/admin/retrieve-data?hl=pt-br#go-go)

dinosaursRef.orderByChild("height").startAt(3).addChildEventListener(new ChildEventListener() {  
  @Override  
  public void onChildAdded(DataSnapshot dataSnapshot, String prevChildKey) {  
    System.out.println(dataSnapshot.getKey());  
  }  
  
  // ...  
});

Você pode usar endAt() para encontrar todos os dinossauros cujos nomes venham, alfabeticamente, antes de pterodáctilo:

[Java](https://firebase.google.com/docs/database/admin/retrieve-data?hl=pt-br#java-java)[Node.js](https://firebase.google.com/docs/database/admin/retrieve-data?hl=pt-br#node.js-node.js)[Python](https://firebase.google.com/docs/database/admin/retrieve-data?hl=pt-br#python-python)[Go](https://firebase.google.com/docs/database/admin/retrieve-data?hl=pt-br#go-go)

dinosaursRef.orderByKey().endAt("pterodactyl").addChildEventListener(new ChildEventListener() {  
  @Override  
  public void onChildAdded(DataSnapshot dataSnapshot, String prevChildKey) {  
    System.out.println(dataSnapshot.getKey());  
  }  
  
  // ...  
});

**startAt()** e **endAt()** são inclusivos, o que significa que "pterodáctilo" corresponderá à consulta acima.

Você pode combinar startAt() e endAt() para limitar as duas extremidades da sua consulta. O seguinte exemplo encontra todos os dinossauros cujos nomes começam com a letra "b":

[Java](https://firebase.google.com/docs/database/admin/retrieve-data?hl=pt-br#java-java)[Node.js](https://firebase.google.com/docs/database/admin/retrieve-data?hl=pt-br#node.js-node.js)[Python](https://firebase.google.com/docs/database/admin/retrieve-data?hl=pt-br#python-python)[Go](https://firebase.google.com/docs/database/admin/retrieve-data?hl=pt-br#go-go)

dinosaursRef.orderByKey().startAt("b").endAt("b\uf8ff").addChildEventListener(new ChildEventListener() {  
  @Override  
  public void onChildAdded(DataSnapshot dataSnapshot, String prevChildKey) {  
    System.out.println(dataSnapshot.getKey());  
  }  
  
  // ...  
});

O caractere **\uf8ff** usado na consulta acima é um ponto de código muito alto no intervalo Unicode. Como ele vem depois da maioria dos caracteres Unicode regulares, a consulta corresponde todos os valores que começam com b.

O método equalTo() permite filtrar com base em correspondências exatas. Como nas outras consultas de intervalo, ele será acionado para cada nó filho correspondente. Por exemplo, você pode usar a seguinte consulta para encontrar todos os dinossauros com 25 metros de altura:

[Java](https://firebase.google.com/docs/database/admin/retrieve-data?hl=pt-br#java-java)[Node.js](https://firebase.google.com/docs/database/admin/retrieve-data?hl=pt-br#node.js-node.js)[Python](https://firebase.google.com/docs/database/admin/retrieve-data?hl=pt-br#python-python)[Go](https://firebase.google.com/docs/database/admin/retrieve-data?hl=pt-br#go-go)

dinosaursRef.orderByChild("height").equalTo(25).addChildEventListener(new ChildEventListener() {  
  @Override  
  public void onChildAdded(DataSnapshot dataSnapshot, String prevChildKey) {  
    System.out.println(dataSnapshot.getKey());  
  }  
  
  // ...  
});

As consultas de intervalo também são úteis quando você precisa paginar seus dados.

Também é possível combinar **orderByValue()** com **startAt()** e **endAt()** para criar consultas de intervalo.

Funcionamento em conjunto

É possível combinar todas essas técnicas para criar consultas complexas. Por exemplo, você pode descobrir o nome do primeiro dinossauro mais baixo que o estegossauro:

[Java](https://firebase.google.com/docs/database/admin/retrieve-data?hl=pt-br#java-java)[Node.js](https://firebase.google.com/docs/database/admin/retrieve-data?hl=pt-br#node.js-node.js)[Python](https://firebase.google.com/docs/database/admin/retrieve-data?hl=pt-br#python-python)[Go](https://firebase.google.com/docs/database/admin/retrieve-data?hl=pt-br#go-go)

dinosaursRef.child("stegosaurus").child("height").addValueEventListener(new ValueEventListener() {  
  @Override  
  public void onDataChange(DataSnapshot stegoHeightSnapshot) {  
    Integer favoriteDinoHeight = stegoHeightSnapshot.getValue(Integer.class);  
    Query query = dinosaursRef.orderByChild("height").endAt(favoriteDinoHeight).limitToLast(2);  
    query.addValueEventListener(new ValueEventListener() {  
      @Override  
      public void onDataChange(DataSnapshot dataSnapshot) {  
        // Data is ordered by increasing height, so we want the first entry  
        DataSnapshot firstChild = dataSnapshot.getChildren().iterator().next();  
        System.out.println("The dinosaur just shorter than the stegosaurus is: " + firstChild.getKey());  
      }  
  
      @Override  
      public void onCancelled(DatabaseError databaseError) {  
        // ...  
      }  
    });  
  }  
  
  @Override  
  public void onCancelled(DatabaseError databaseError) {  
    // ...  
  }  
});

Ordenação dos dados

Esta seção explica a ordenação dos seus dados usando cada uma das quatro funções de ordenação.

orderByChild

Ao usar orderByChild(), os dados que contêm a chave filha especificada são ordenados desta maneira:

1. Filhos com valor null para a chave filha especificada vêm primeiro.
2. Filhos com valor false para a chave filha especificada vêm em seguida. Se vários filhos tiverem o valor false, eles serão classificados [lexicograficamente](http://en.wikipedia.org/wiki/Lexicographical_order) por chave.
3. Filhos com valor true para a chave filha especificada vêm depois. Se vários filhos tiverem um valor true, eles serão ordenados alfabeticamente, por chave.
4. Filhos com um valor numérico são os próximos, ordenados em ordem crescente. Se vários filhos tiverem o mesmo valor numérico para o nó filho especificado, eles serão ordenados por chave.
5. As strings vêm depois dos números e são classificadas alfabeticamente em ordem crescente. Se vários filhos tiverem o mesmo valor para o nó filho especificado, eles serão ordenados alfabeticamente, por chave.
6. Os objetos vêm por último e são ordenados alfabeticamente, por chave, em ordem crescente.

orderByKey

Quando usamos orderByKey() para ordenar os dados, eles são retornados em ordem ascendente, por chave, da seguinte maneira. Lembre-se de que as chaves só podem ser strings.

1. Filhos com uma chave que possa ser analisada como um número inteiro de 32 bits vêm primeiro, ordenados em ordem crescente.
2. Filhos com um valor de string como chave são os próximos, ordenados alfabeticamente em ordem crescente.

orderByValue

Ao usar orderByValue(), os filhos são ordenados pelo valor deles. Os critérios de ordenação são os mesmos de orderByChild(), com a exceção de que o valor usado **é o do nó e não o de uma chave filha especificada**.