100 XP

Describe types of non-relational data

4 minutes

Non-relational data generally falls into two categories; semi-structured and non-structured. In this unit, you'll learn about what these terms mean, and see some examples.

What is semi-structured data?

Semi-structured data is data that contains fields. The fields don't have to be the same in every entity. You only define the fields that you need on a per-entity basis. The Customer entities shown in the previous unit are examples of semi-structured data. The data must be formatted in such a way that an application can parse and process it. One common way of doing this is to store the data for each entity as a JSON document. The term *JSON* stands for JavaScript Object Notation; it's the format used by JavaScript applications to store data in memory, but can also be used to read and write documents to and from files.

A JSON document is enclosed in curly brackets ({ and }). Each field has a name (a label), followed by a colon, and then the value of the field. Fields can contain simple values, or subdocuments (each starting and ending with curly brackets). Fields can also have multiple values, held as arrays and surrounded with square brackets ([and]). Literals, or fixed values, in a field are enclosed in quotes, and fields are separated with commas.

The example below shows the customers from the previous unit, formatted as JSON documents:

JSON Copy

```json
{
  "ID": "1",
  "Name": "Mark Hanson",
  "Telephone": [
    { "Home": "1-999-9999999" },
    { "Business": "1-888-8888888" },
    { "Cell": "1-777-7777777" }
  ],
  "Address": [
    { "Home": [
      { "StreetAddress": "121 Main Street" },
      { "City": "Some City" },
      { "State": "NY" },
      { "Zip": "10110" }
    ] },
    { "Business": [
      { "StreetAddress": "87 Big Building" },
      { "City": "Some City" },
      { "State": "NY" },
      { "Zip": "10111" }
    ] }
  ]
}


{
  "ID": "2",
  "Title": "Mr",
  "Name": "Jeff Hay",
  "Telephone": [
    { "Home": "0044-1999-333333" },
    { "Mobile": "0044-17545-444444" }
  ],
  "Address": [
    { "UK": [
      { "StreetAddress": "86 High Street" },
      { "Town": "Some Town" },
      { "County": "A County" },
      { "Postcode": "GL8888" },
      { "Region": "UK" }
    ] },
    { "US": [
      { "StreetAddress": "777 7th Street" },
      { "City": "Another City" },
      { "State": "CA" },
      { "Zip": "90111" }
    ] }
  ]
}
```

You're free to define whatever fields you like. The important point is that the data follows the JSON grammar. When an application reads a document, it can use a JSON parser to break up the document into its component fields and extract the individual pieces of data.

Other formats you might see include *Avro*, *ORC*, and *Parquet*:

- *Avro* is a row-based format. It was created by Apache. Each record contains a header that describes the structure of the data in the record. This header is stored as JSON. The data is stored as binary information. An application uses the information in the header to parse the binary data and extract the fields it contains. Avro is a very good format for compressing data and minimizing storage and network bandwidth requirements. This example is a subset of the header information for the previous example, formatted as Avro:

Avro Copy

```
{
  "type": "record",
  "name": "contact_schema",
  "fields": [
   {
      "name": "id",
      "type": "int",
      "doc": "ID of the contact"
    },
    {
      "name": "name",
      "type": "string",
      "doc": "Name of the contact"
    },
{
      "name": "telephone",
      "type": [
        "null",
        {
          "type": "array",
          "items": {
            "type": "record",
            "name": "contact_schema.telephone",
            "fields": [
              {
                "name": "phoneid",
                "type": "int"
              },
              {
                "name": "phonetype",
                "type": [ "null", "string" ]
              }
            ]
          }
        }
      ]
    }
  ]
}
```

- *ORC* (Optimized Row Columnar format) organizes data into columns rather than rows. It was developed by HortonWorks for optimizing read and write operations in Apache Hive. Hive is a data warehouse system that supports fast data summarization and querying over very large datasets. Hive supports SQL-like queries over unstructured data. An ORC file contains *stripes* of data. Each stripe holds the data for a column or set of columns. A stripe contains an index into the rows in the stripe, the data for each row, and a footer that holds statistical information (count, sum, max, min, and so on) for each column.

- *Parquet* is another columnar data format. It was created by Cloudera and Twitter. A Parquet file contains row groups. Data for each column is stored together in the same row group. Each row group contains one or more chunks of data. A Parquet file includes metadata that describes the set of rows found in each chunk. An application can use this metadata to quickly locate the correct chunk for a given set of rows, and retrieve the data in the specified columns for these rows. Parquet specializes in storing and processing nested data types efficiently. It supports very efficient compression and encoding schemes.

What is unstructured data?

Unstructured data is data that doesn't naturally contain fields. Examples include video, audio, and other media streams. Each item is an amorphous blob of binary data. You can't search for specific elements in this data.

You might choose to store data such as this in storage that is specifically designed for the purpose. In Azure, you would probably store video and audio data as block blobs in an Azure Storage account. (The term *blob* stands for Binary Large Object*). A block blob only supports basic read and write operations.

You could also consider files as a form of unstructured data, although in some cases a file might include metadata that indicates what type of file it is (photograph, Word document, Excel spreadsheet, and so on), owner, and other elements that could be stored as fields. However, the main content of the file is unstructured.

Next unit: Describe types of non-relational and NoSQL databases

Continue >