Specify nested and repeated columns in table schemas

This page describes how to define a table schema with nested and repeated columns in BigQuery. For an overview of table schemas, see Specifying a schema (/bigquery/docs/schemas).

Define nested and repeated columns

To create a column with nested data, set the data type of the column to `RECORD` in the schema. A `RECORD` can be accessed as a `STRUCT` (/bigquery/docs/reference/standard-sql/data-types#struct_type) type in standard SQL. A `STRUCT` is a container of ordered fields.

To create a column with repeated data, set the mode (/bigquery/docs/schemas#modes) of the column to `REPEATED` in the schema. A repeated field can be accessed as an `ARRAY` (/bigquery/docs/reference/standard-sql/data-types#array_type) type in standard SQL.

A `RECORD` column can have `REPEATED` mode, which is represented as an array of `STRUCT` types. Also, a field within a record can be repeated, which is represented as a `STRUCT` that contains an `ARRAY`. An array cannot contain another array directly. For more information, see Declaring an `ARRAY` type (/bigquery/docs/reference/standard-sql/data-types#declaring_an_array_type).

Limitations

Nested and repeated schemas are subject to the following limitations:

A schema cannot contain more than 15 levels of nested `RECORD` types.

Columns of type `RECORD` can contain nested `RECORD` types, also called *child* records. The maximum nested depth limit is 15 levels. This limit is independent of whether the `RECORD`s are scalar or array-based (repeated).

Example schema

The following example shows sample nested and repeated data. This table contains information about people. It consists of the following fields:

- `id`
- `first_name`
- `last_name`
- `dob` (date of birth)
- `addresses` (a nested and repeated field)
  - `addresses.status` (current or previous)
  - `addresses.address`
  - `addresses.city`
  - `addresses.state`
  - `addresses.zip`
  - `addresses.numberOfYears` (years at the address)

The JSON data file would look like the following. Notice that the addresses column contains an array of values (indicated by `[ ]`). The multiple addresses in the array are the repeated data. The multiple fields within each address are the nested data.

```
{"id":"1","first_name":"John","last_name":"Doe","dob":"1968-01-22","addresses":[{"status":"current","address":"123 F
{"id":"2","first_name":"Jane","last_name":"Doe","dob":"1980-10-16","addresses":[{"status":"current","address":"789 Ar
```

The schema for this table looks like the following:

```
[
    {
        "name": "id",
        "type": "STRING",
        "mode": "NULLABLE"
    },
    {
        "name": "first_name",
        "type": "STRING",
        "mode": "NULLABLE"
    },
    {
        "name": "last_name",
        "type": "STRING",
        "mode": "NULLABLE"
    },
    {
        "name": "dob",
        "type": "DATE",
        "mode": "NULLABLE"
    },
    {
        "name": "addresses",
        "type": "RECORD",
        "mode": "REPEATED",
        "fields": [
            {
                "name": "status",
                "type": "STRING",
                "mode": "NULLABLE"
            },
            {
                "name": "address",
                "type": "STRING",
                "mode": "NULLABLE"
            },
            {
                "name": "city",
                "type": "STRING",
                "mode": "NULLABLE"
            },
            {
                "name": "state",
                "type": "STRING",
                "mode": "NULLABLE"
            },
            {
                "name": "zip",
                "type": "STRING",
                "mode": "NULLABLE"
            },
            {
                "name": "numberOfYears",
                "type": "STRING",
                "mode": "NULLABLE"
            }
        ]
    }
]
```

Specifying the nested and repeated columns in the example

Console (#console)bq (#bq)Go (#go)Java (#java)Node.js (#node.js)Python (#python)

Before trying this sample, follow the Python setup instructions in the BigQuery quickstart using client libraries (/bigquery/docs/quickstarts/quickstart-client-libraries). For more information, see the BigQuery Python API reference documentation (https://googleapis.dev/python/bigquery/latest/index.html).

View on GitHub (https://github.com/googleapis/python-bigquery/blob/35627d145a41d57768f19d4392ef235928e00f72/docs/snippets.py)

```
# from google.cloud import bigquery
# client = bigquery.Client()
# project = client.project
# dataset_ref = bigquery.DatasetReference(project, 'my_dataset')

schema = [
    bigquery.SchemaField("id", "STRING", mode="NULLABLE"),
    bigquery.SchemaField("first_name", "STRING", mode="NULLABLE"),
    bigquery.SchemaField("last_name", "STRING", mode="NULLABLE"),
    bigquery.SchemaField("dob", "DATE", mode="NULLABLE"),
    bigquery.SchemaField(
        "addresses",
        "RECORD",
        mode="REPEATED",
        fields=[
            bigquery.SchemaField("status", "STRING", mode="NULLABLE"),
            bigquery.SchemaField("address", "STRING", mode="NULLABLE"),
            bigquery.SchemaField("city", "STRING", mode="NULLABLE"),
            bigquery.SchemaField("state", "STRING", mode="NULLABLE"),
            bigquery.SchemaField("zip", "STRING", mode="NULLABLE"),
            bigquery.SchemaField("numberOfYears", "STRING", mode="NULLABLE"),
        ],
    ),
]
table_ref = dataset_ref.table("my_table")
table = bigquery.Table(table_ref, schema=schema)
table = client.create_table(table)  # API request

print("Created table {}".format(table.full_table_id))
```

Modify nested and repeated columns

After you add a nested column or a nested and repeated column to a table's schema definition, you can modify the column as you would any other type of column. BigQuery natively supports several schema changes such as adding a new nested field to a record or relaxing a nested field's mode. For more information, see Modifying table schemas (/bigquery/docs/managing-table-schemas).

Additionally, you can manually modify a schema definition that includes nested and repeated columns. For more information, see Manually changing table schemas (/bigquery/docs/manually-changing-schemas).

When to use nested and repeated columns

BigQuery performs best when your data is denormalized. Rather than preserving a relational schema such as a star or snowflake schema, denormalize your data and take advantage of nested and repeated columns. Nested and repeated columns can maintain relationships without the performance impact of preserving a relational (normalized) schema.

For example, a relational database used to track library books would likely keep all author information in a separate table. A key such as `author_id` would be used to link the book to the authors.

In BigQuery, you can preserve the relationship between book and author without creating a separate author table. Instead, you create an author column, and you nest fields within it such as the author's first name, last name, date of birth, and so on. If a book has multiple authors, you can make the nested author column repeated.

BigQuery supports loading nested and repeated data from source formats that support object-based schemas, such as JSON files, Avro files, Firestore export files, and Datastore export files.

Table security

To control access to tables in BigQuery, see Introduction to table access controls (/bigquery/docs/table-access-controls-intro).

Next steps

- To insert and update rows with nested and repeated columns, see Data manipulation language syntax (/bigquery/docs/reference/standard-sql/dml-syntax).

Last updated 2021-08-17 UTC.