[544] BigQuery: Complex Types

Tyler Caraza-Harter

Learning Objectives

- authenticate on a GCP VM to gain access to BiqQuery datasets

- execute BiqQuery queries in a variety of settings (console, Jupyter with extension, Python call)

- query complex data types (arrays and structs) using correlated cross joins

Demos...

Outline

Types

Basics

- BOOL, INT64, FLOAT64
- STRING, BYTES
- DATE, DATETIME
- etc.

Nesting

- **ARRAY** (repeated):
  `myarray[OFFSET(5)]`
- **STRUCT** (record)
  `mystruct.some_attribute`

example from https://cloud.google.com/bigquery/docs/nested-repeated

```
+--------------------+-----------------------------------+------------+
| title              | authors         array of structs  | num_pages  |
+--------------------+-----------------------------------+------------+
| Example Book One   | [{123, Alex, 01-01-1960},         | 487        |
|                    |  {789, Kim, 01-01-1980}]          |            |
| Example Book Two   | [{456, Rosario, 01-01-1970}]      | 89         |
+--------------------+-----------------------------------+------------+
                                      struct
```

Outline

Types: Simple and Arrays/Structs

Cross Joining

Unnesting, Correlated Cross Join

Geographic Data

Cross Joins

Previously covered JOIN types:
- INNER, LEFT, RIGHT

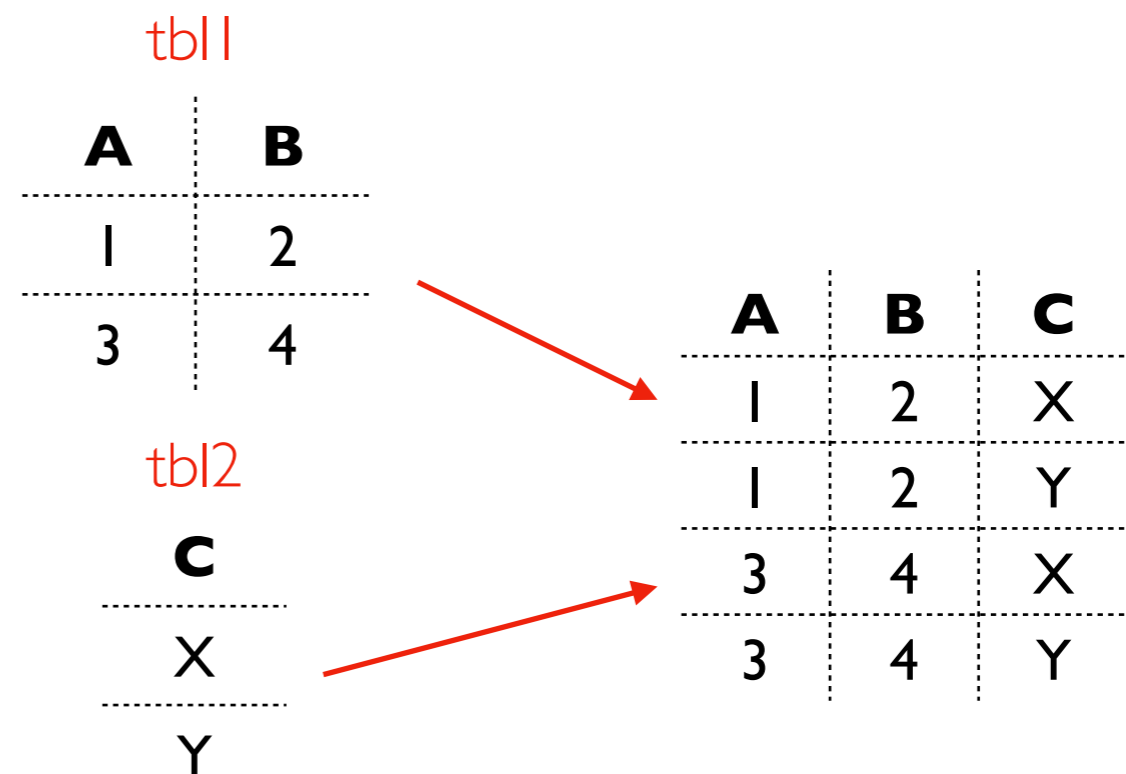CROSS JOIN: every row in table 1 with every row in table 2

format 1

SELECT *
FROM **tbl1**
CROSS JOIN **tbl2**

format 2

SELECT *
FROM **tbl1**, **tbl2**

same meaning as format 1
(comma means "cross join")

tbl1

A	B
1	2
3	4

tbl2

C
X
Y

A	B	C
1	2	X
1	2	Y
3	4	X
3	4	Y

Cross Joins: Filtering

Predicates

- don't use "ON" as in other JOINs
- can optionally use "WHERE"

Naive version: get every combination of pairs, then filter down after.
Can we do better?

Sometimes query engines can optimize certain WHERE filters with CROSS JOIN.

BigQuery implements optimized spatial JOINs for INNER JOIN and CROSS JOIN operators with the following GoogleSQL predicate functions:

- `ST_DWithin`
- `ST_Intersects`
- `ST_Contains`
- `ST_Within`
- `ST_Covers`
- `ST_CoveredBy`
- `ST_Equals`
- `ST_Touches`

BigQuery Documentation

Outline

Types: Simple and Arrays/Structs

Cross Joining

Unnesting, Correlated Cross Join

Geographic Data

Unnesting and Correlated Cross Join

SELECT x,y,z
FROM **tbl**
CROSS JOIN UNNEST(**tbl.coord)**

x	coord
1	[{2,3}, {4,5}]
6	[{7,8}, {9,10}]

unnest

y	z
2	3
4	5

different logical table for each row

y	z
7	8
9	10

Unnesting and Correlated Cross Join

SELECT x,y,z
FROM **tbl**
CROSS JOIN UNNEST(**tbl.coord)**

x	coord
1	[{2,3}, {4,5}]
6	[{7,8}, {9,10}]
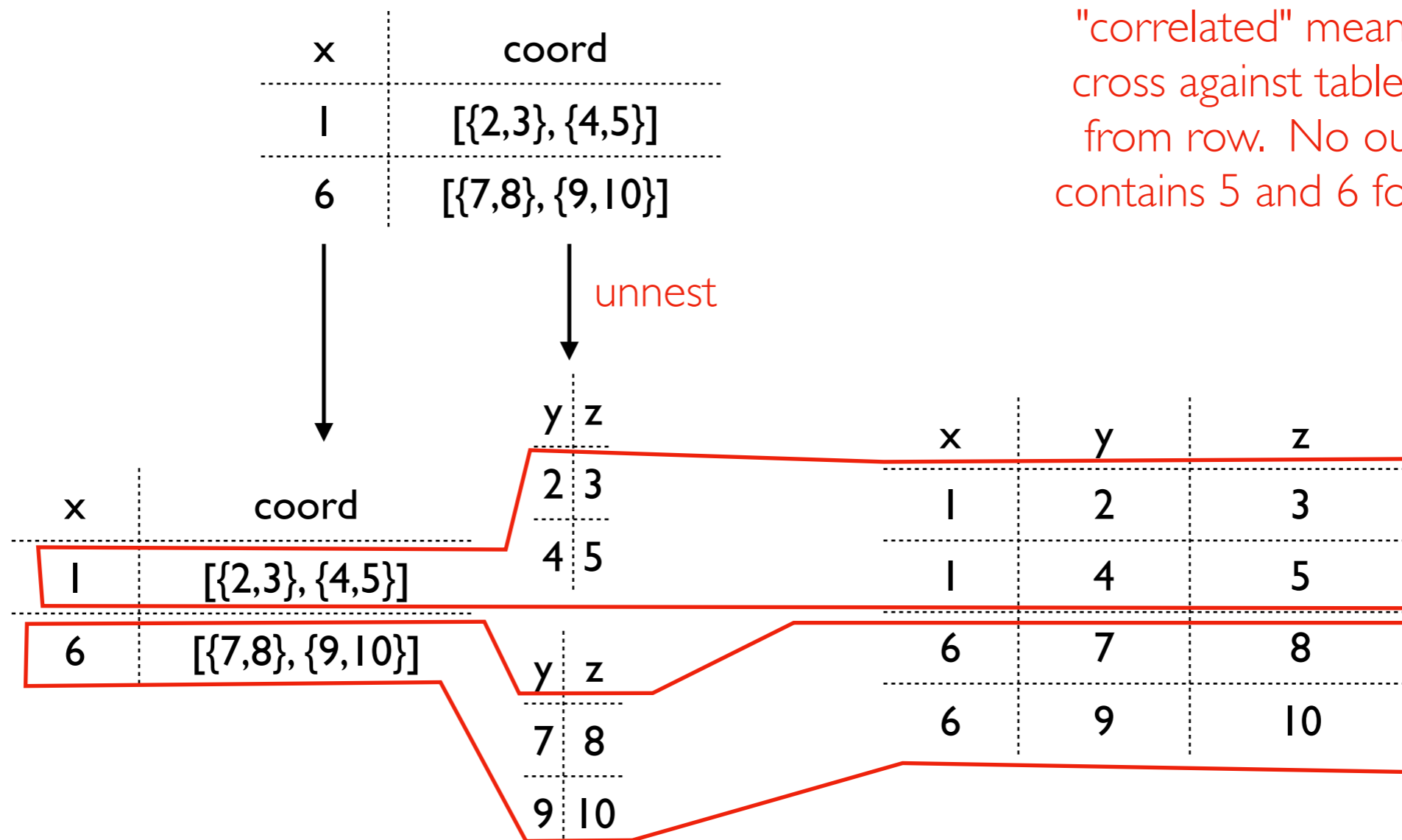
unnest

"correlated" means we only cross against table unnested from row. No output row contains 5 and 6 for example.

x	coord
1	[{2,3}, {4,5}]
6	[{7,8}, {9,10}]

y	z
2	3
4	5

y	z
7	8
9	10

x	y	z
1	2	3
1	4	5
6	7	8
6	9	10

TopHat

Outline

Types: Simple and Arrays/Structs

Cross Joining

Unnesting, Correlated Cross Join

Geographic Data
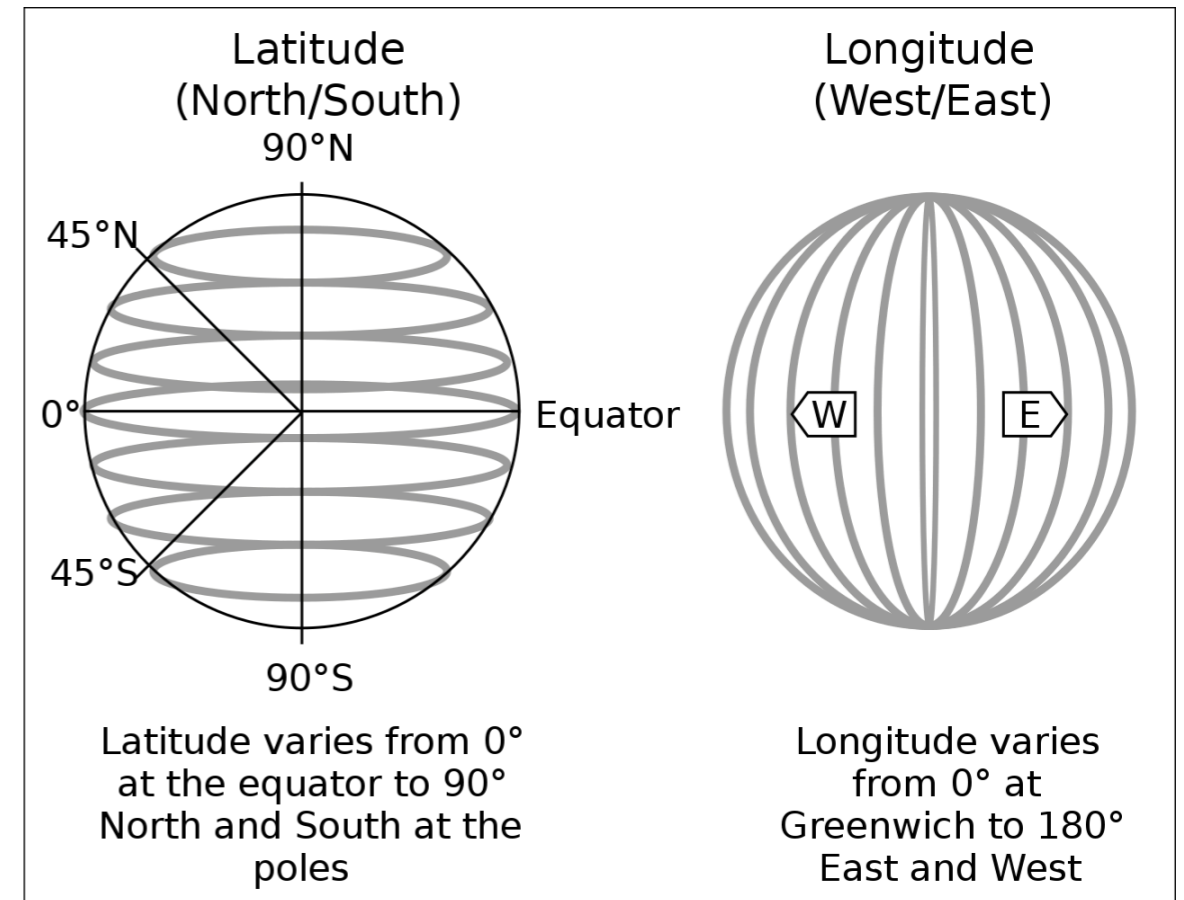
Geographic Data

Coordinate reference systems

- way to associate coordinates with a point on earth
- latitude/longitude (used by GPS) is most famous
- some systems incorporate altitude too (3D coordinate system)

BigQuery support

- common geo operations (e.g., geographic joins)
- uses lat/lon by default (no altitude)

Shape constructors

- ST_GEOGPOINT ●
- ST_MAKELINE
- ST_MAKEPOLYGON △

Latitude (North/South)	Longitude (West/East)
90°N	
45°N	
0° ... Equator	W E
45°S	
90°S	
Latitude varies from 0° at the equator to 90° North and South at the poles	Longitude varies from 0° at Greenwich to 180° East and West

https://en.wikipedia.org/wiki/
Geographic_coordinate_system#/media/
File:FedStats_Lat_long.svg

other shape (e.g., multi-polygons) are possible with operations on these

Demos...