What is a column family database?

NOSQL CONCEPTS

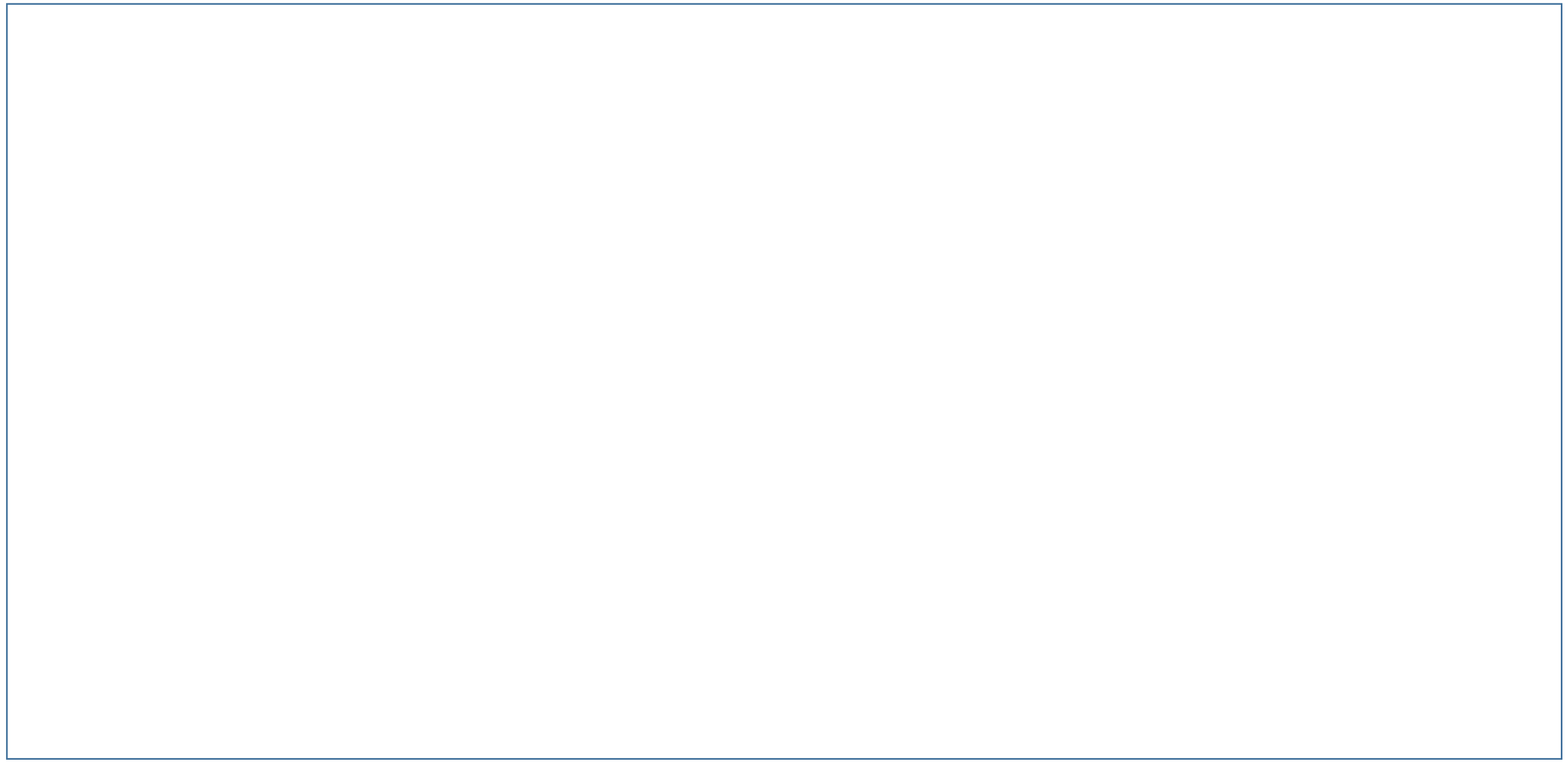
Miriam Antona
Software engineer

Column family databases - overview

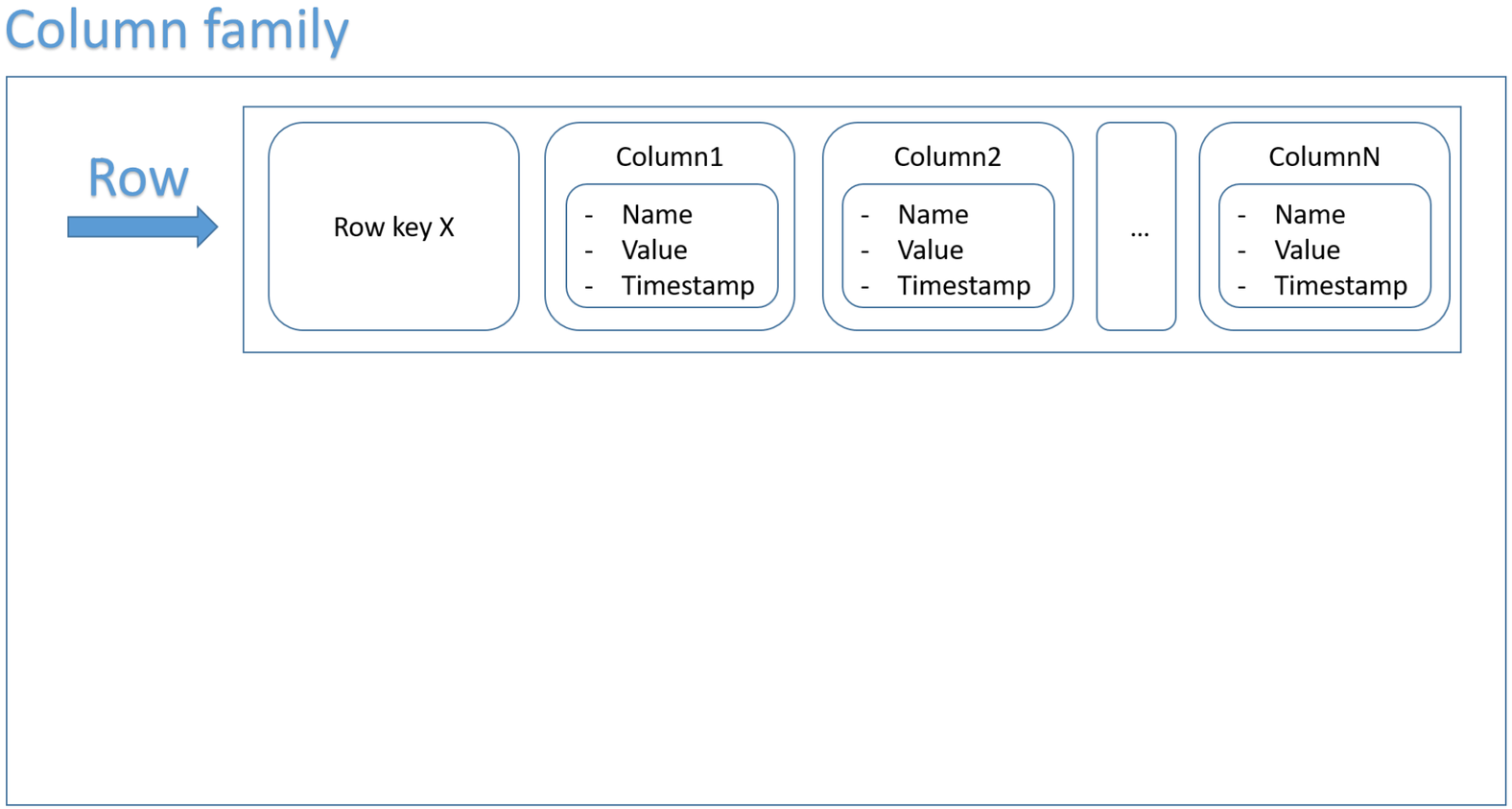
- Derive from **Google BigTable**
- Store data in **column families**
  - group related data
  - frequently accessed together
- Also called **wide column** databases
- Great when dealing with large volumes of data

Column family databases - structure

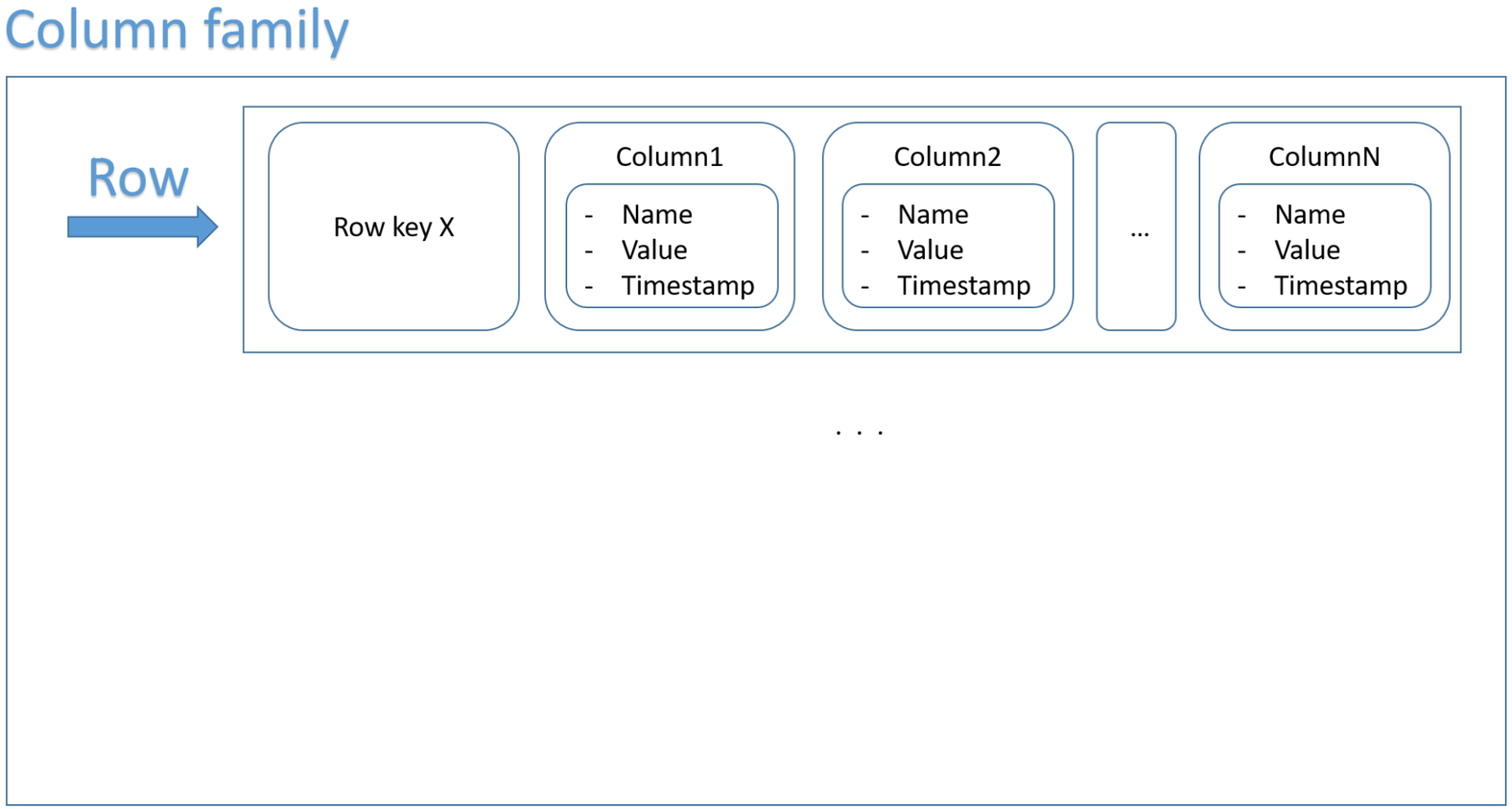
Column family



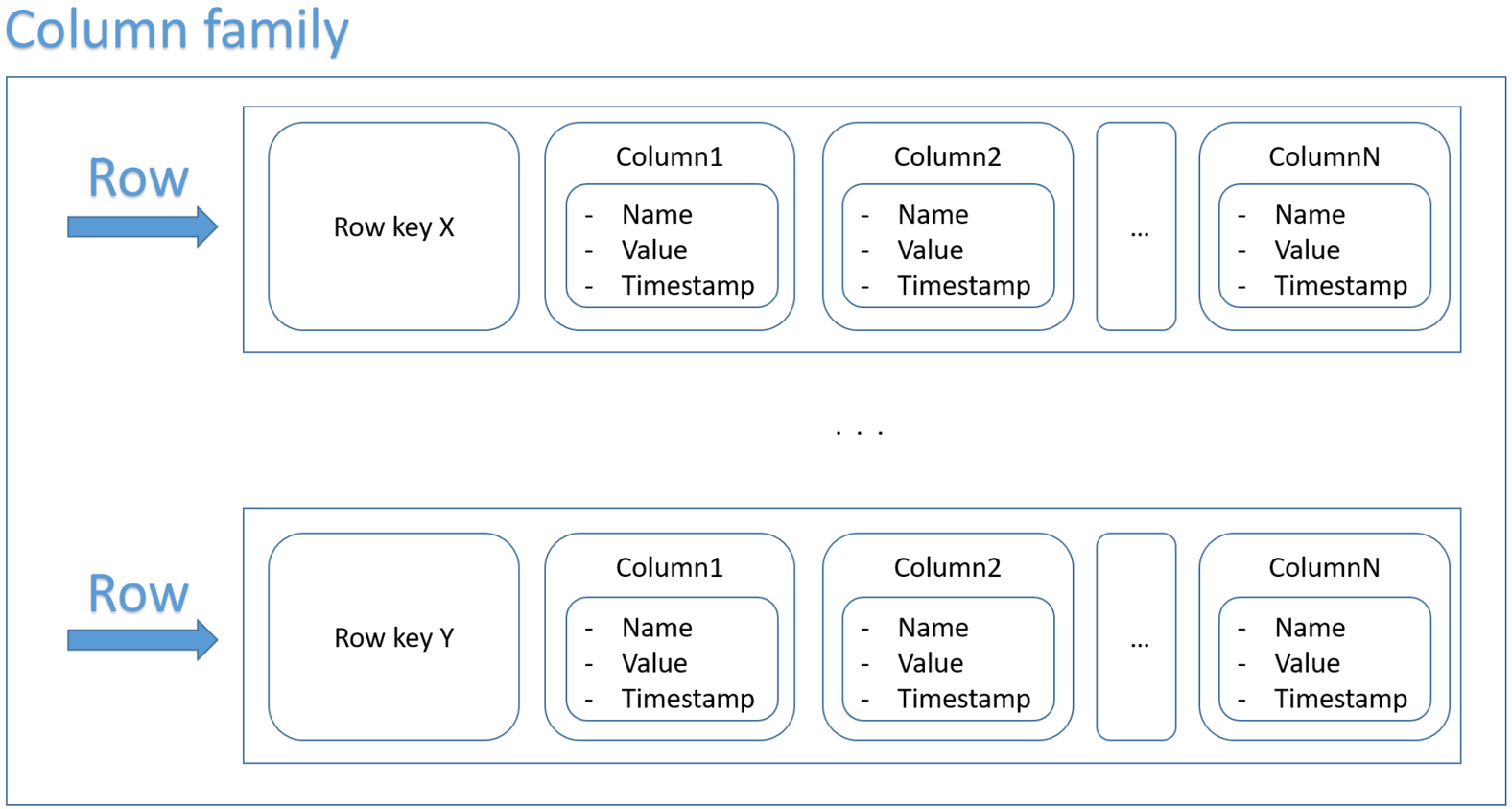
Column family databases - structure



Column family databases - structure

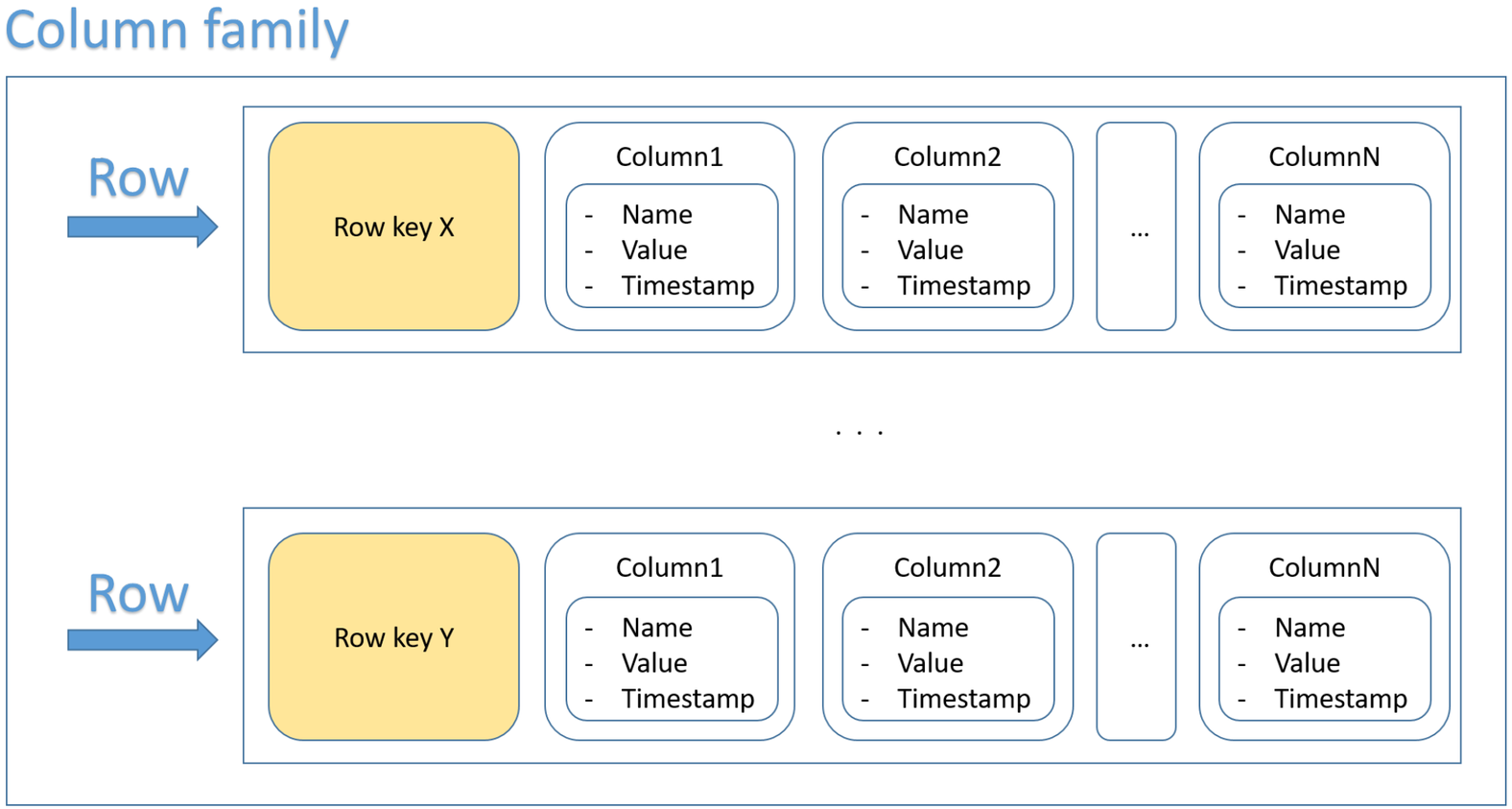


Column family databases - structure



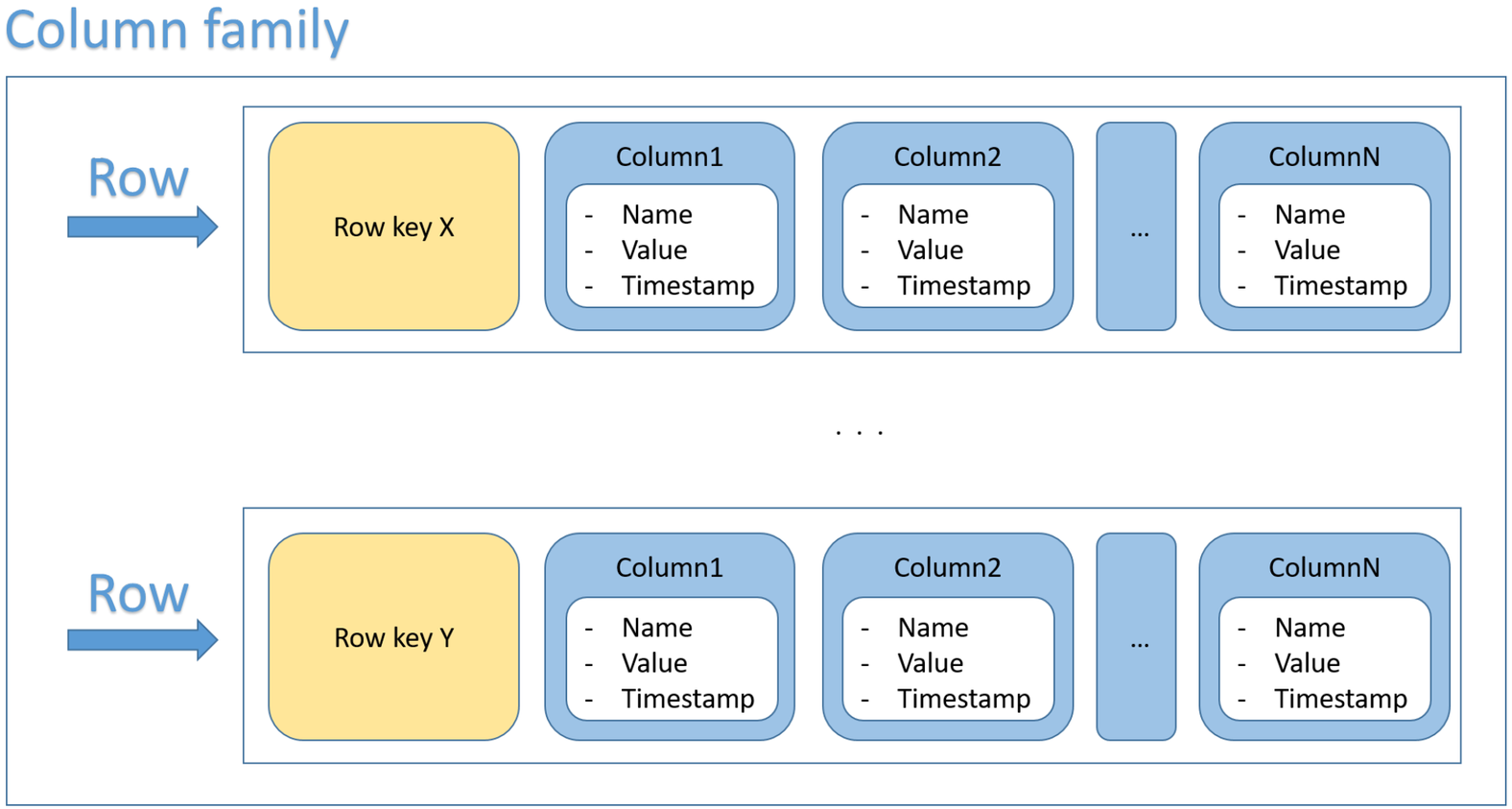
- A **column family** is like a **table** in a relational database

Column family databases - structure



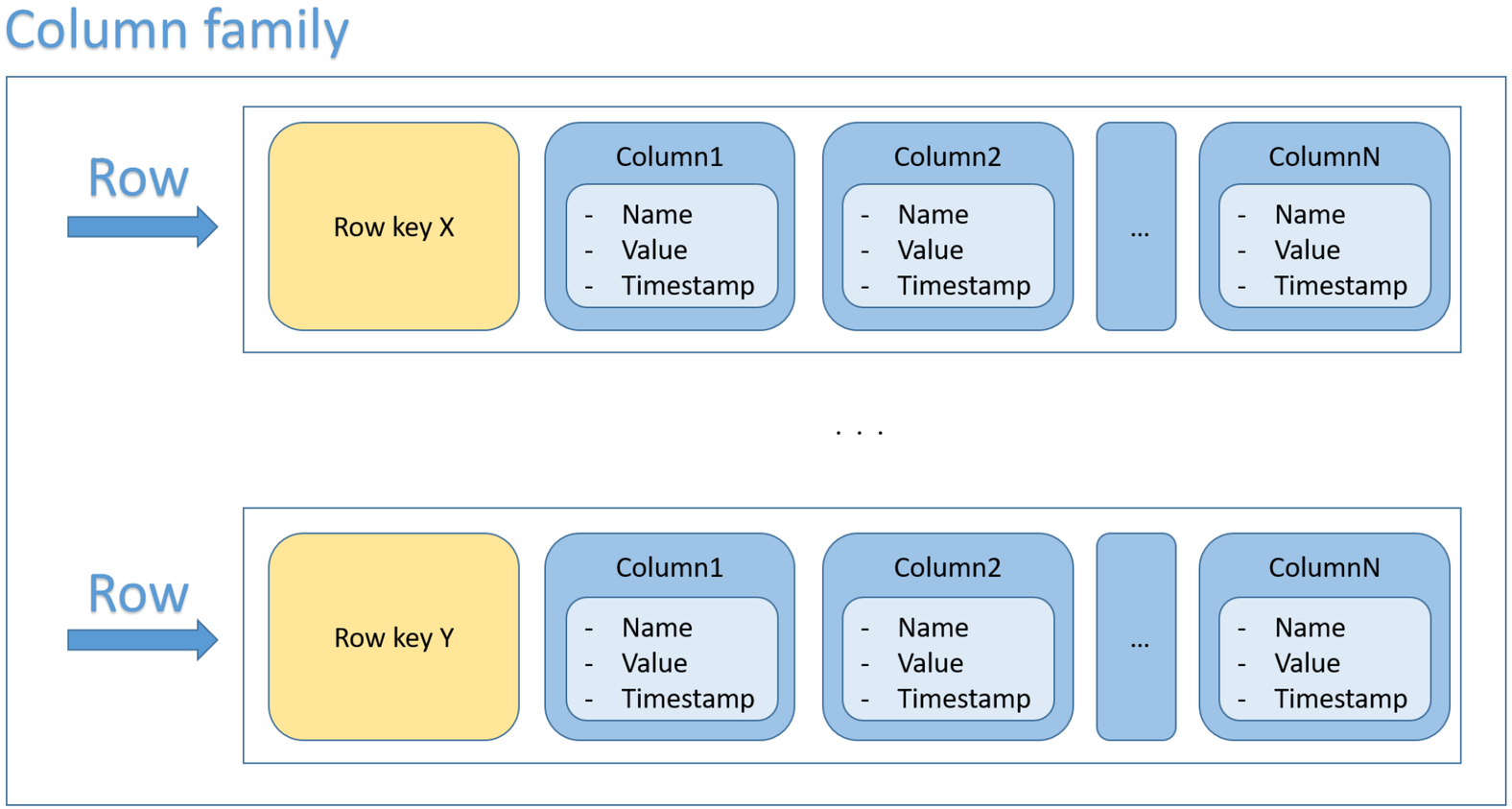
- **Row key**: unique identifiers
  - Like primary keys in a relational database

Column family databases - structure



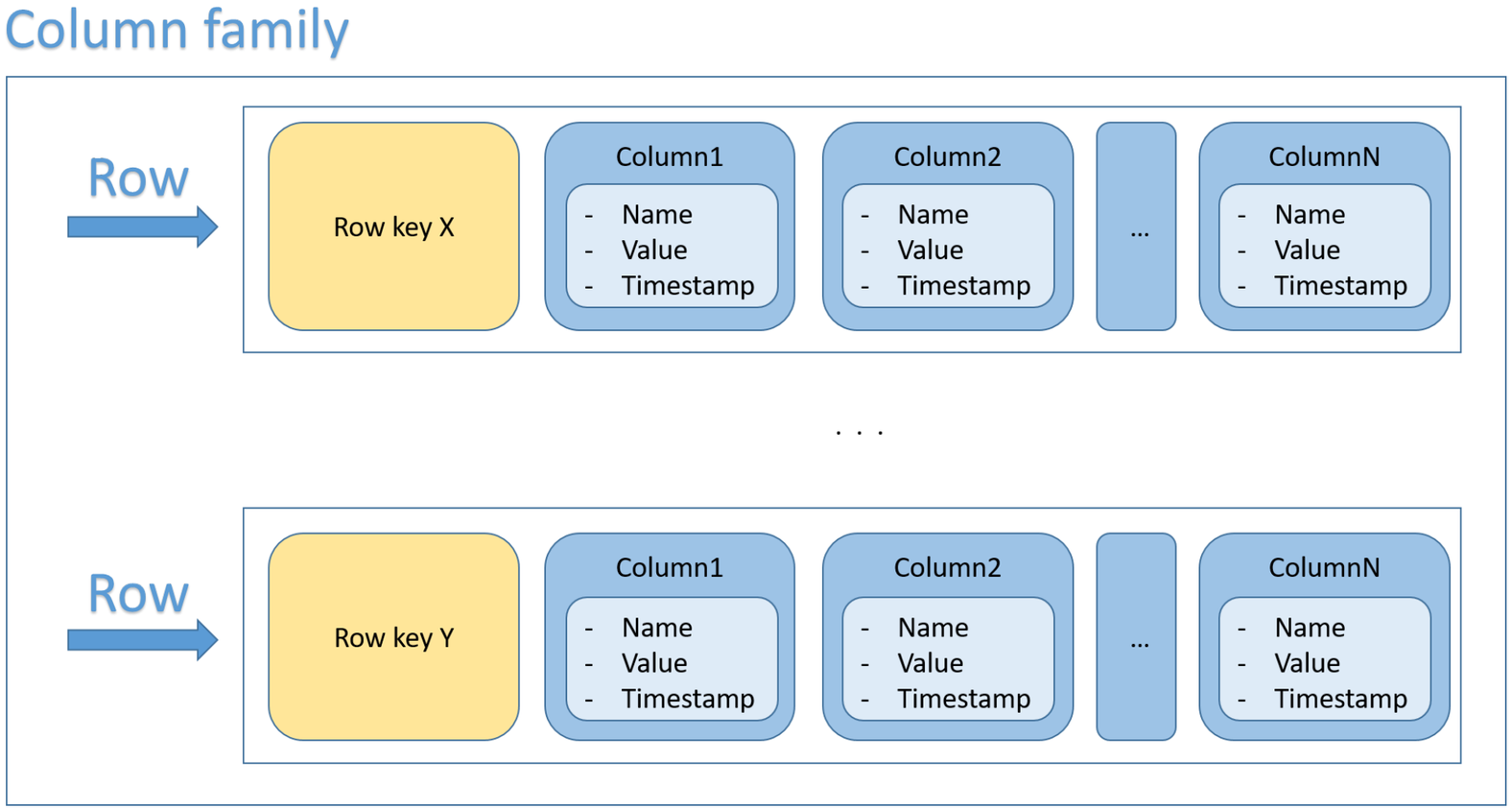
- Each **row** can have different number of columns
  - Columns can be added when needed

Column family databases - structure



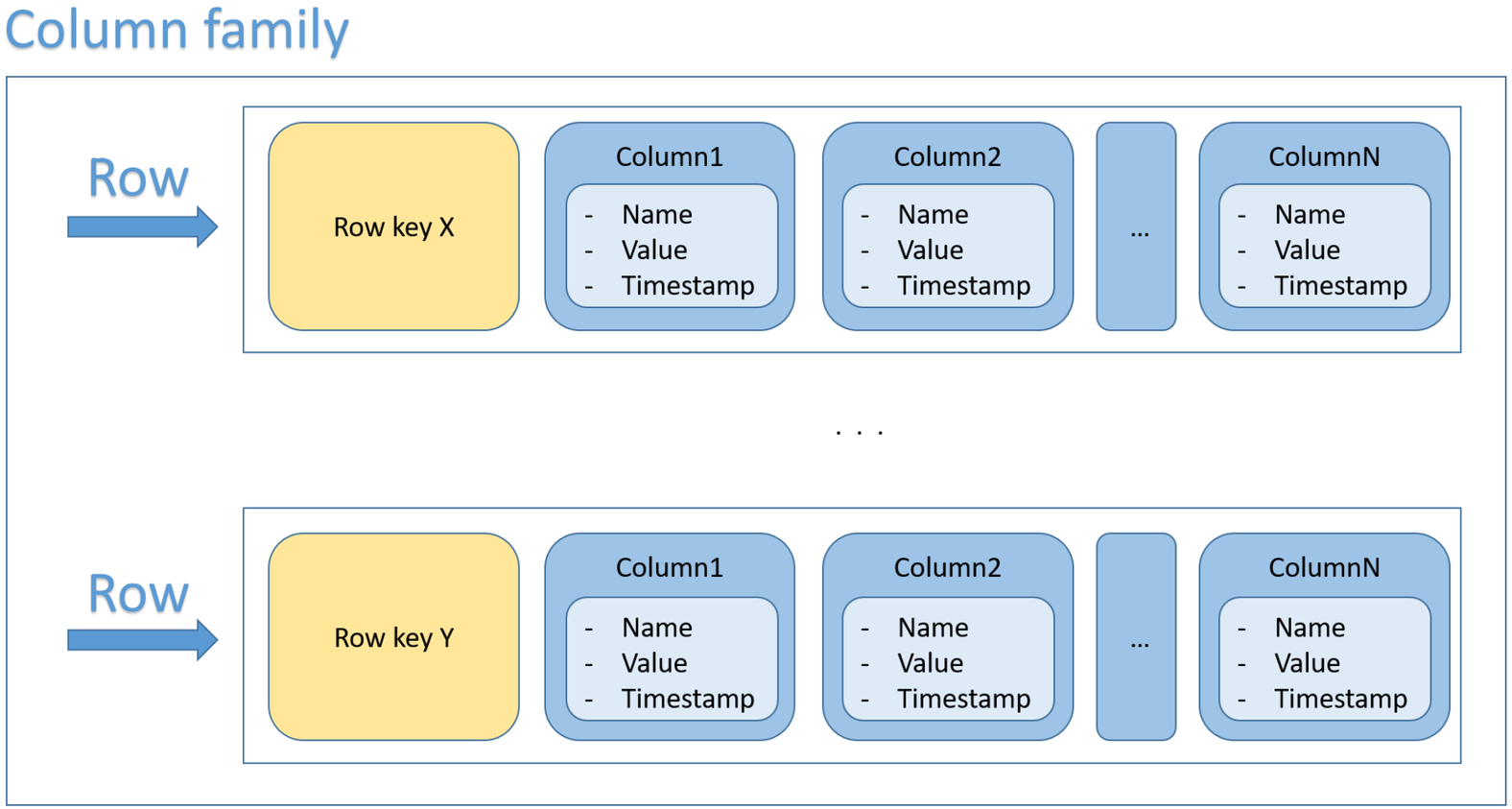
- **Parts** of the columns:
  - Name, value, and timestamp

Column family databases - structure



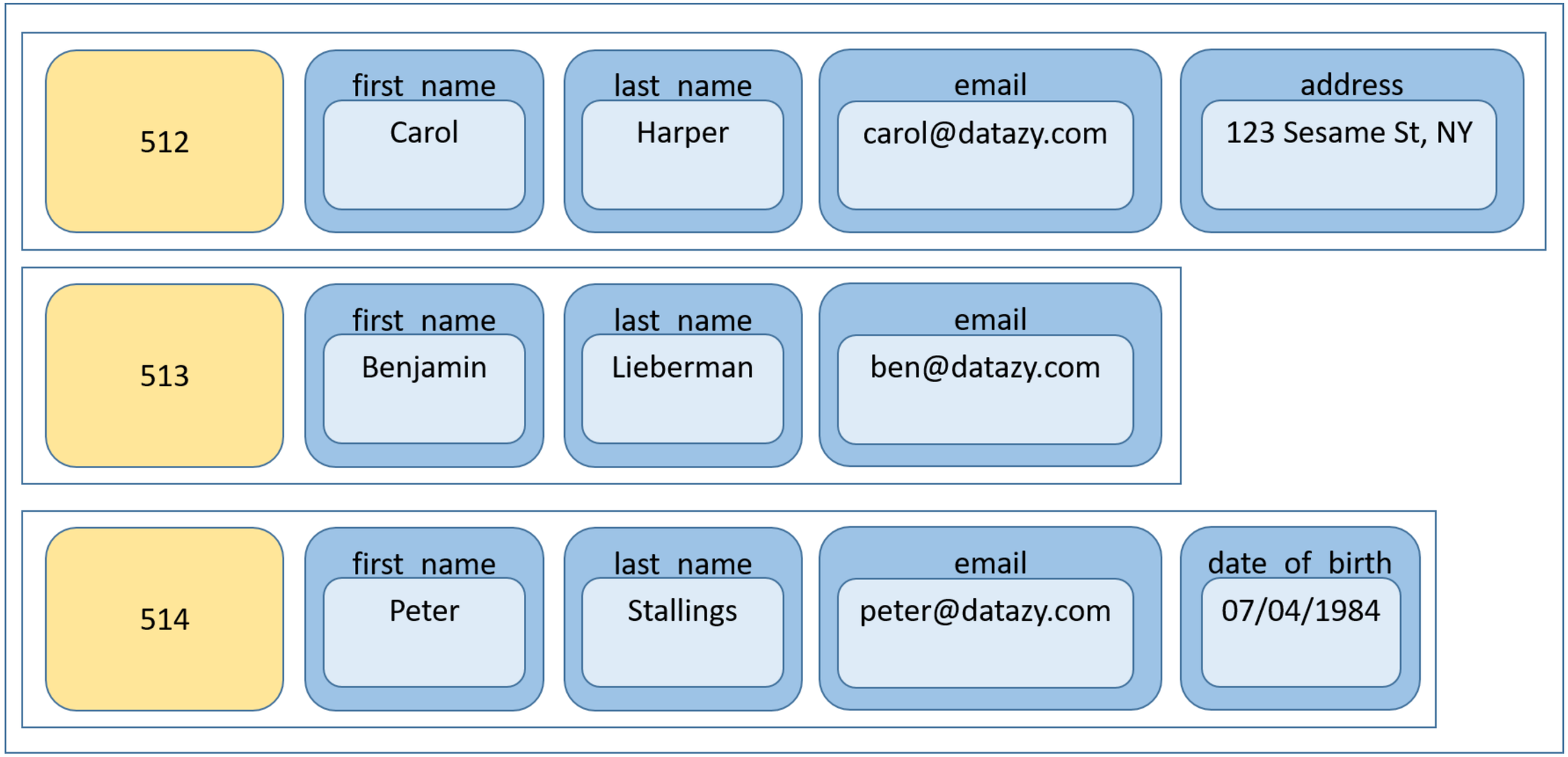
- **Value:** specify the type depending on the database

Column family databases - structure



- **Timestamps:** store date and time when the data was inserted.
  - Multiple values of a column

Column family databases - example



Column family databases - designing

- Think about the queries
- No joins
  - Add all the columns we need

Popular column family databases

Let's practice!

NOSQL CONCEPTS

Advantages and limitations of column family databases

NOSQL CONCEPTS

Miriam Antona
Software engineer

Advantages - flexibility

- Rows within a column family can have **different columns**
- **Add new columns** to a row if we need them
- **Avoids** filling with **default values**
- Flexibility mustn't be considered as the only criterion
  - Evaluate key-value and document databases

Advantages - speed

- Related columns are **stored together on disk**
- **Very fast** writing / retrieving

Advantages - scalability

- Scale horizontally
  - Sharding across multiple servers

Advantages - large volumes of data

- Designed to handle **large volumes of data**
  - speed
  - horizontal scalability
  - efficient data compression

Limitations

- Atomic reads/writes but **no multirow transactions**
- **No joins** support
- **No subqueries** support
- Need to **define the queries** quite well
  - Queries change -> may need to change the column families
  - Can be costly

Let's practice!

NOSQL CONCEPTS

When to use column family databases

NOSQL CONCEPTS

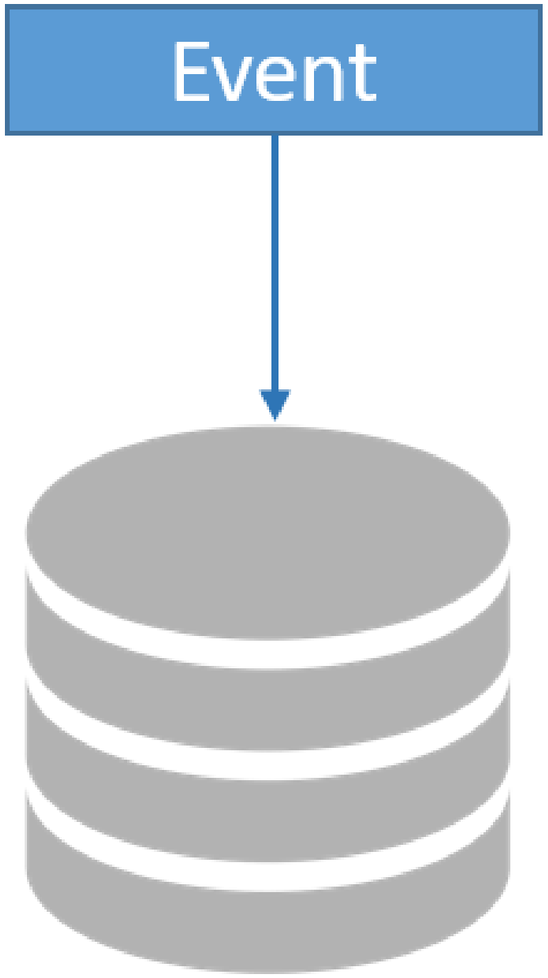
Miriam Antona
Software engineer

Suitable cases - general cases

- Large volumes of data
- Extreme write speeds

Suitable cases - event logging

- Types of events:
  - User logging
  - Errors
  - ...

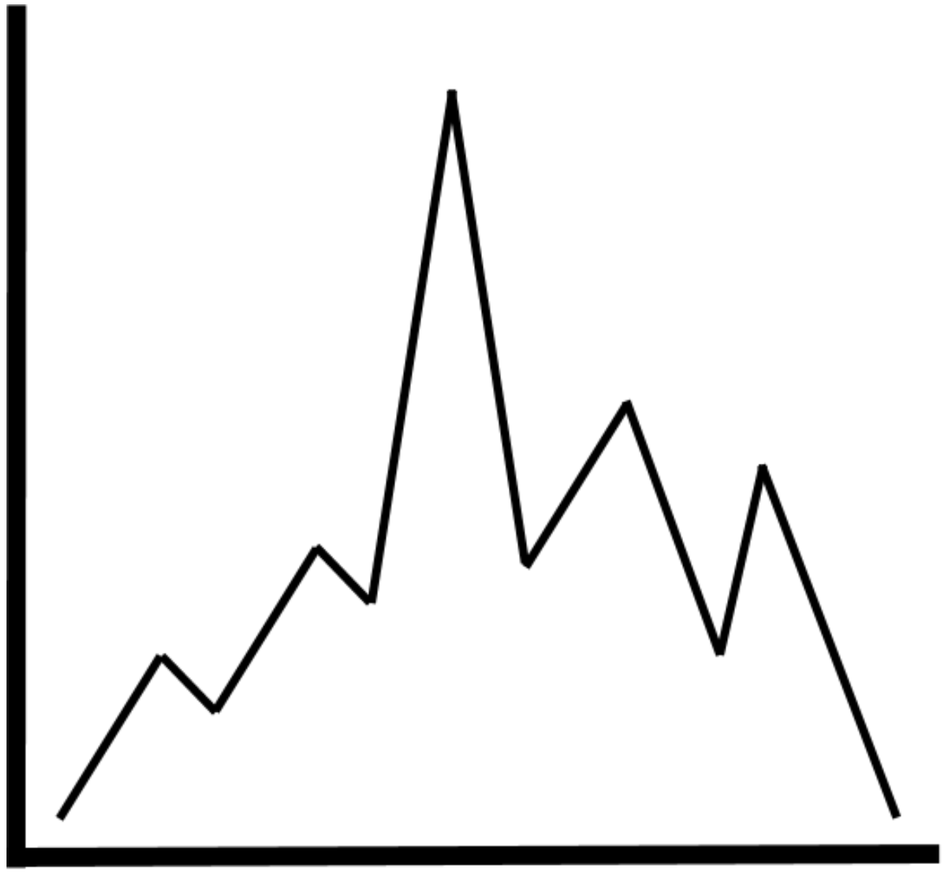


Suitable cases - Content Management Systems

- Comments
- Links
- Tags
- ...

Suitable cases - time-series data

- Weather
- Traffic
- etc.



Unsuitable cases

- **Prototyping** and at the **beginning** of a project
  - Need to change the queries very frequently
  - Changing the queries -> may imply changing the design of the column families
  - Costly and may slow down the productivity
- Complex queries and joins
- Not dealing with large amounts of data

Let's practice!

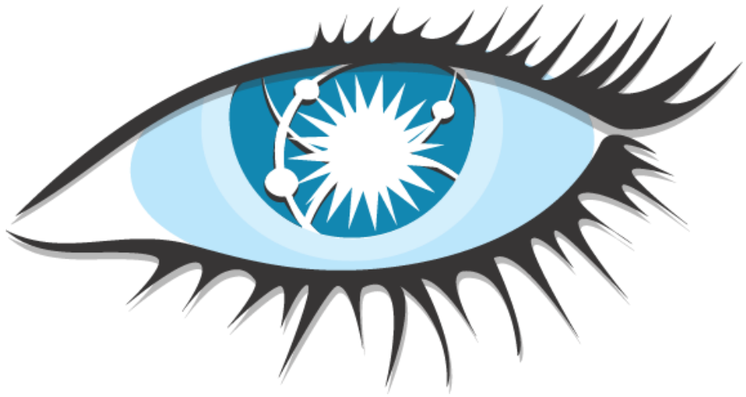
NOSQL CONCEPTS

Apache Cassandra case study

NOSQL CONCEPTS

Miriam Antona
Software engineer

Apache Cassandra - overview



Apache CASSANDRA

- Popular **column family** database
- Originally developed by **Facebook**
- **Open-source**
- Finally became a project of the **Apache Foundation**

Apache Cassandra - features

- **Distributed**
  - Data is distributed across the nodes of the cluster
  - Every node plays the same role
  - No master node
- **High availability**
- **No single point of failure**
- **Scales horizontally** by adding nodes
- Cassandra client **drivers**: C#, Java, Python, Scala, etc.

Apache Cassandra - features

- Cassandra Query Language (aka **CQL**)
  - Query data
  - Similar syntax to SQL
  - Tables (for column families), rows, and columns
  - Differences between CQL and SQL:
    - no joins
    - no foreign keys
    - no subqueries, etc.
    - rows can contain a different number of columns

```
SELECT * FROM users WHERE user_id IN (212, 213, 214);
```

Apache Cassandra - ecosystem

- **Third-party** Cassandra projects, tools, products, and services
  - Cloud offerings
  - Installation tools
  - Developers' frameworks
  - Connectors
  - etc.

Apache Cassandra - customers

Bigmate case study - overview

- Location tracking
- Industrial sensor
- Productivity

Bigmate case study - problem and solution

- **IoT platform:**
  - Ingests and processes **large volumes** of **different data**
  - **Integrate** IoT sensors, devices, and other platforms
  - Process data in **real-time**
  - Scale and deploy across **multiple locations**
  - Application examples:
    - Thermy -> capture the skin temperature of people
    - Warny -> detects possible collisions
- Tested MySQL, MongoDB, Apache Cassandra, etc.
  - Chose **Apache Cassandra**
  - **Scaled better**

Bigmate case study - results

- **Millions of operations** of **concurrent** users
- Display 20,000 real-time data points to a single customer
- Fault tolerance (data replication)

Bigmate case study - results

- **Millions of operations** of **concurrent** users
- Display 20,000 real-time data points to a single customer
- Fault tolerance (data replication)

[1] https://cassandra.apache.org/case-studies/

Let's practice!

NOSQL CONCEPTS