Zed: Leveraging Data Types to Process Eclectic Data

Amy Ousterhout, Steve McCanne, Henri Dubois-Ferriere, Silvery Fu, Sylvia Ratnasamy, Noah Treuhaft

UC Berkeley, Brim Data, 12th Ave Labs

The Rise of Eclectic Data

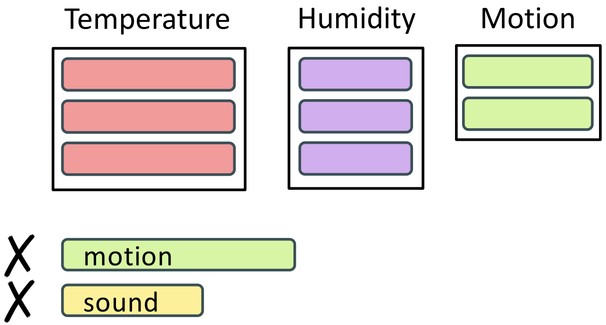
- Eclectic data:
  - Heterogeneous – spanning many schemas
  - Evolving – schemas change over time
- Increasingly common due to:
  - IoT, monitoring
  - Relating independent data sets
- Poses new challenges for ingestion, storage, querying, and introspection

what schemas are present?
what fields does each have?
etc.

Existing Data-Processing Approaches

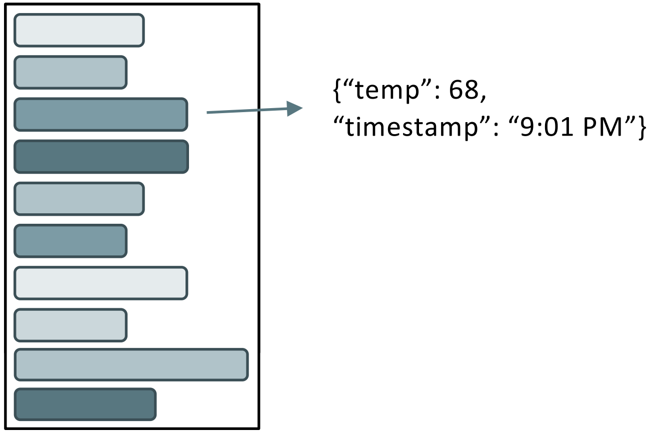
- Two common approaches today: the relational model and document model

Relational Model



- ✓ efficient storage and querying
- ✓ some support for data introspection
- ✗ difficult to handle changing schemas

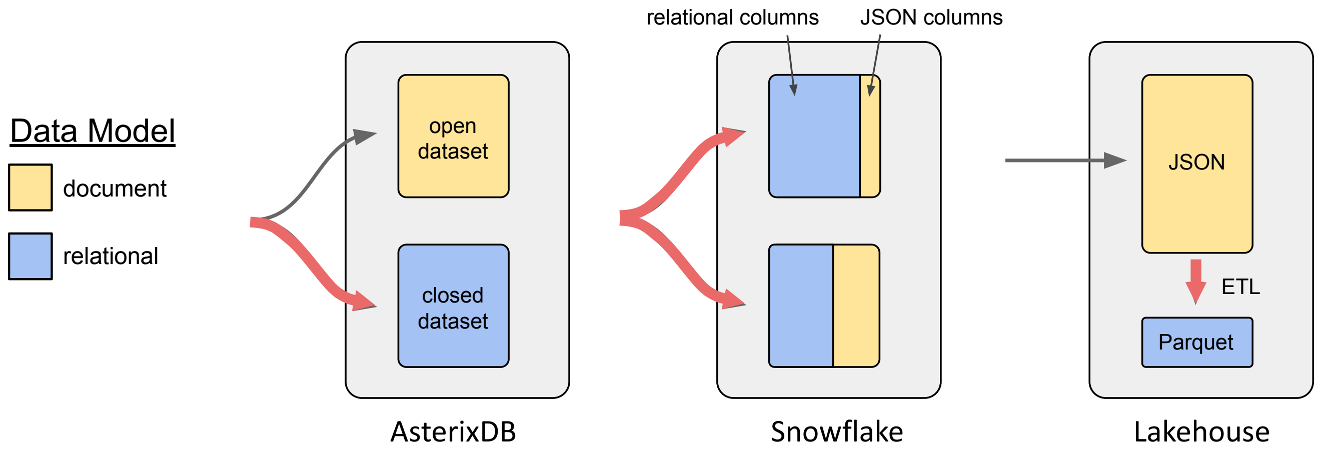
Document Model (e.g., JSON)



{“temp”: 68,
“timestamp”: “9:01 PM”}

- ✗ inefficient storage and querying
- ✗ no support for data introspection
- ✓ easy to mix eclectic data

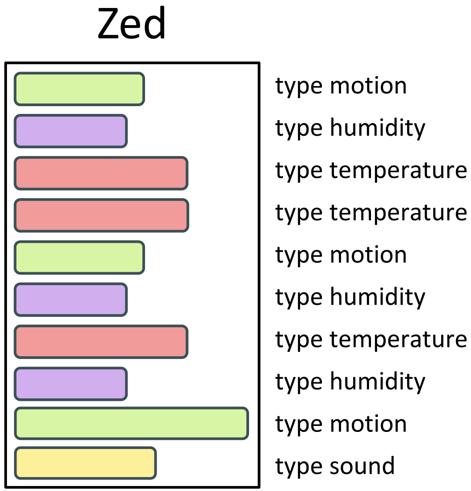
The Limitations of Hybrid Approaches



- Still require cleaning data into the relational model
- Users must decide which model(s) to use for each piece of data
- Limited introspection
  - Only in the relational model
  - No holistic way to refer to schemas

Zed: A Unified Approach

- Goals:
  - Unify the relational and document models, embodying both *simultaneously* ✓
  - Enable data introspection ✓
- Requirements:
  - Specify the *complete type* of each piece of data
  - Be *flexible* about which types of data can coexist
- Key idea: new *data type abstraction*
  - Associated directly with individual data values
  - First-class – holistic way to refer to types

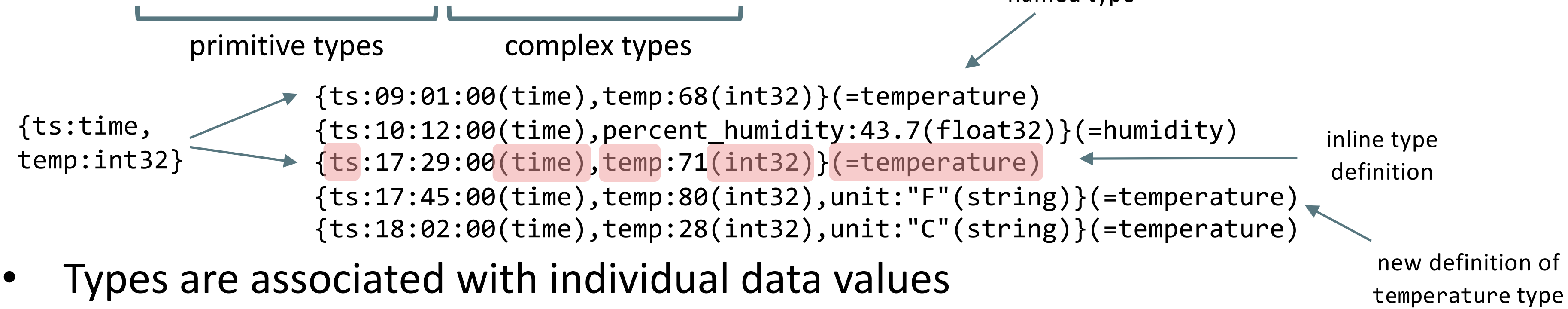


Zed Components and Design Questions

- Data model
  - Should types be open or closed? Partial or complete?
  - What should the scope of a type definition be?
- Query language
  - What operators are necessary for data introspection?
- Family of data formats
  - How to represent type information?

Zed Data Model

- Ordered sequence of typed data values
  - Int32, string, bool, record, array, set, etc.



primitive types

complex types

named type

{ts:time,
temp:int32}

```
{ts:09:01:00(time),temp:68(int32)}(=temperature)
{ts:10:12:00(time),percent_humidity:43.7(float32)}(=humidity)
{ts:17:29:00(time),temp:71(int32)}(=temperature)
{ts:17:45:00(time),temp:80(int32),unit:"F"(string)}(=temperature)
{ts:18:02:00(time),temp:28(int32),unit:"C"(string)}(=temperature)
```

inline type definition

new definition of temperature type

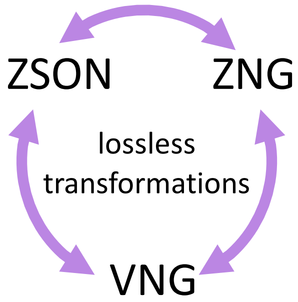
- Types are associated with individual data values
- First-class types (type type)
- Type definitions are stored inline
- Types are complete and closed
- Type definitions may change in a data stream

Zed Query Language

- Subsumes query languages for the relational and document models
- Enables data introspection
- Key new features:
  - Type introspection
    - Obtain the type of an individual data value with `typeof()`
  - First-class types
    - Functions can return types – `typeof()` returns a type (e.g., `<ts:time,temp:int32}>`)
    - Types can be arguments to functions – `is(<temperature>)`
    - Types can be tested for equality – `typeof(this)==<temperature>`
    - Support for type literals – e.g., `<ts:time,temp:int32}>`

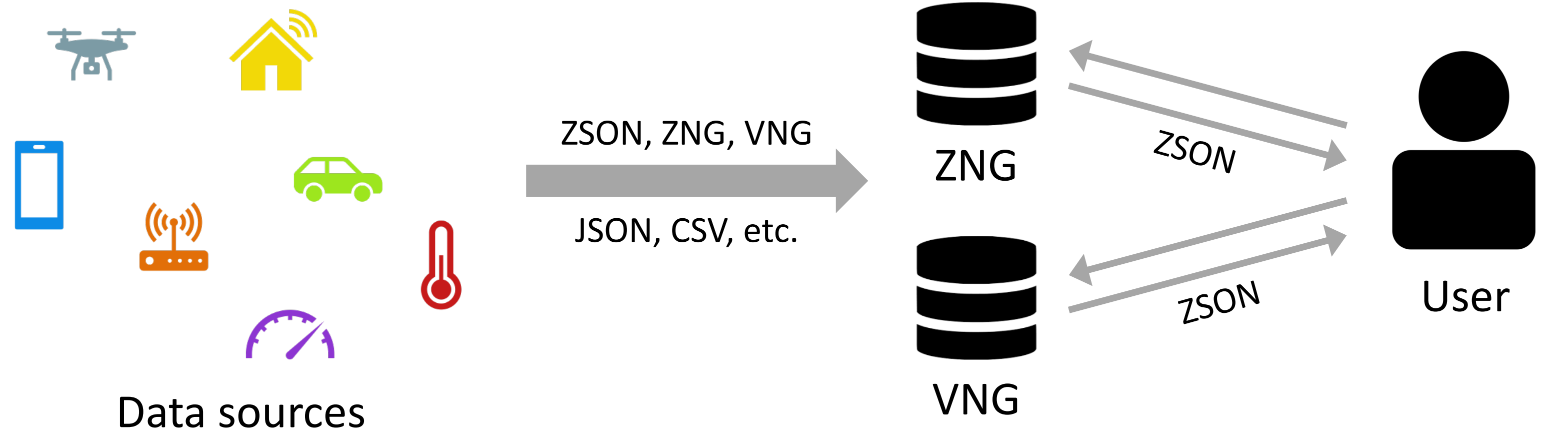
Zed Format Family

- No single format is best for all uses
- Zed provides a family of formats
  - ZSON: text-based
  - ZNG: binary row-based
  - VNG: binary vector, generalizes existing columnar formats
- Lossless transformations between formats
- Binary formats encode types efficiently, once per file



Data Processing with Zed

- Generate data in Zed formats or other formats
- Store in ZNG, VNG, and indexes



Data sources

ZSON, ZNG, VNG

JSON, CSV, etc.

ZNG

VNG

ZSON

ZSON

User

Querying and Introspection in Zed

- Querying – supports search and analytics

```
$ zq -f table "count() by temp" sensor_data.vng
temp   count
68     29
71     82
80     41
...
```

first-class types in the query language

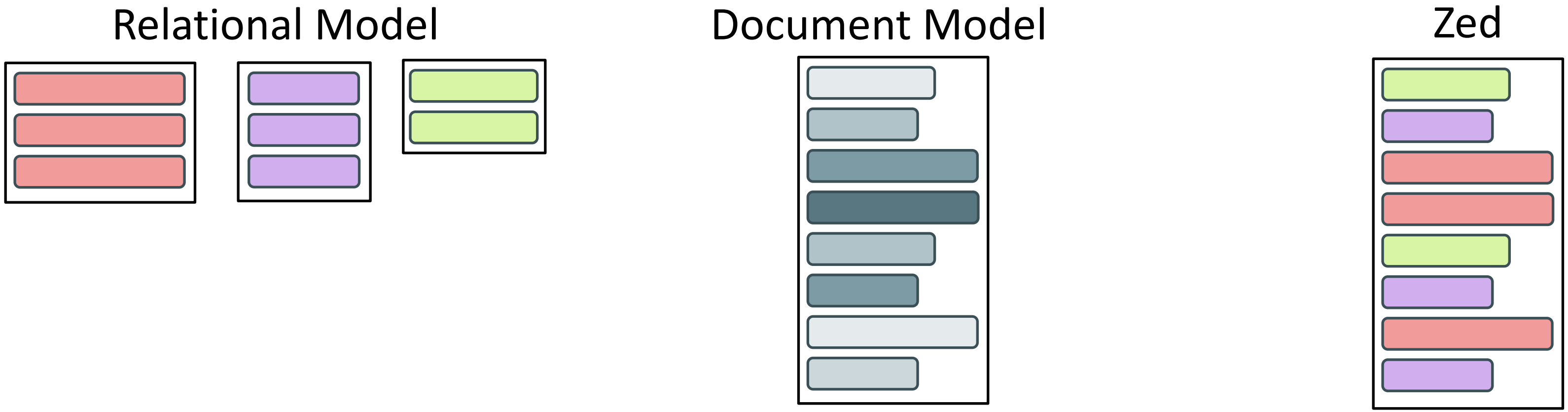
- Introspection

```
$ zq -f table "count() by typeof(this)" sensor_data.vng
typeof                                              count
<temperature={ts:time,temp:int32}>                 452
<humidity={ts:time,percent_humidity:float32}> 82
...
```

first-class types in the data model

Conclusion

- Zed: a new unified approach to data processing
  - Designed to support eclectic data
  - Centered around data types
- Work on Zed is ongoing
- Available open source at: `https://github.com/brimdata/zed`


Relational Model
Document Model
Zed