Building Event Collection SDKs and Data Models

OSA Con ‘22
Paul Boocock
@paul_boocock

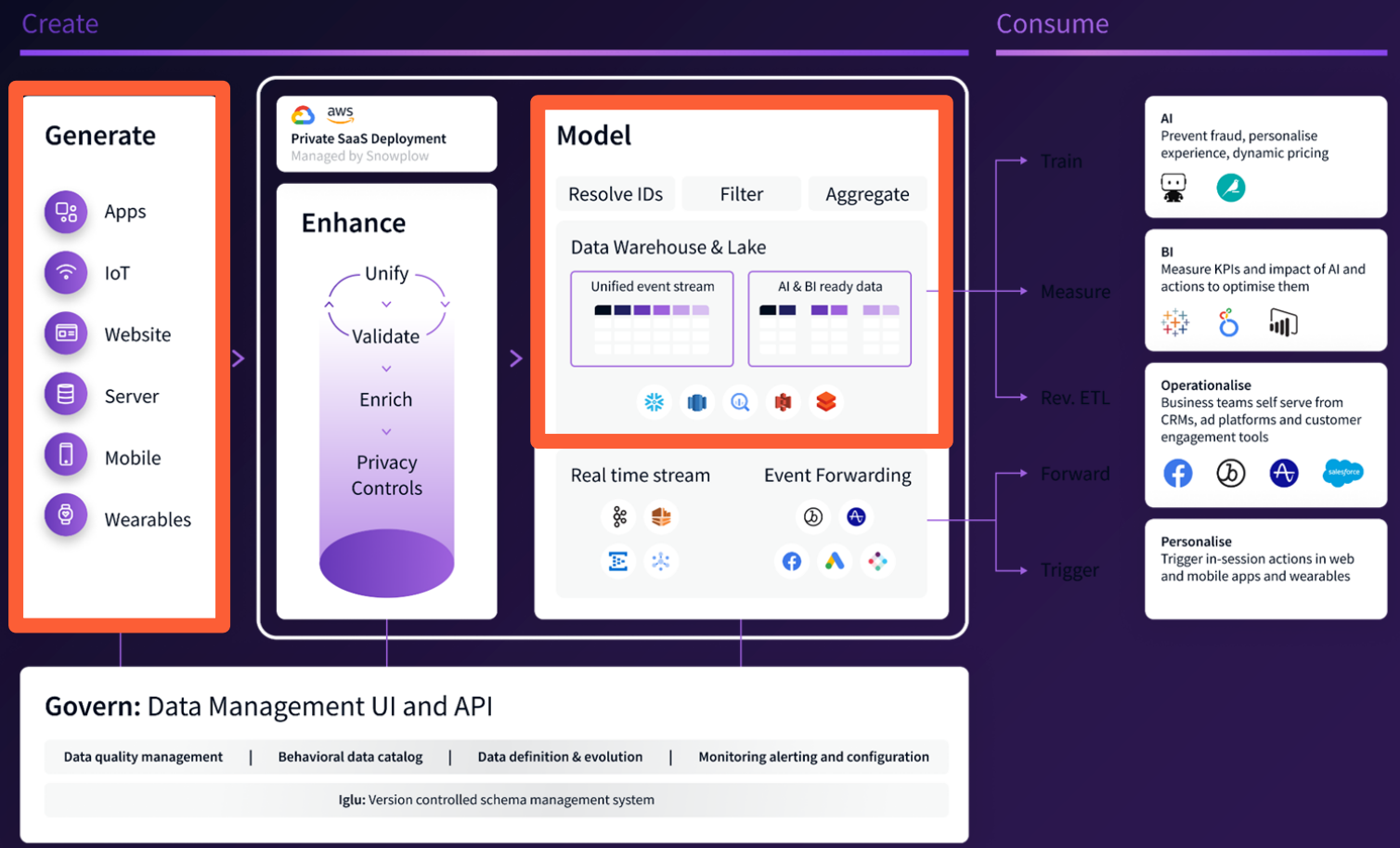
Table of contents

_01

What is Snowplow?

A quick intro

Snowplow: we build tech to enable companies to **Create Data**



_02

Trackers

But first, a small detour…

```
{
                "$schema":
"http://iglucentral.com/schemas/com.snowplowana
lytics.self-desc/schema/jsonschema/1-0-0#",
                "description": "JSON schema for
a button click  event",
                "self": {
                                "vendor":
"com.acme",
                                "name": "click",
                                "format":
"jsonschema",
                                "version": "1-0-
0"
                },

                "type": "object",
                "properties": {
                                "button": {
                                  "type":
["string", "null"],
                                  "maxLength":
255
                                }
                },
                "additionalProperties": false
}
```

The data created looks a bit like this

Event

Default

- user_id
- timestamp
- page_url
- mkt_campaign
- device
- city

Custom

- button_click

Entity

Custom

- content_title
- author
- date_created
- content_id

But can crucially be evolved over time

```
{
                "$schema":
"http://iglucentral.com/schemas/com.snowplowana
lytics.self-desc/schema/jsonschema/1-0-0#",
                "description": "JSON schema for
a button click  event",
                "self": {
                                "vendor":
"com.acme",
                                "name": "click",
                                "format":
"jsonschema",
                                "version": "1-0-
0"
                },

                "type": "object",
                "properties": {
                                "button": {
                                  "type":
["string", "null"],
                                  "maxLength":
255
                                }
                },
                "additionalProperties": false
}
```

```
{
                "$schema":
"http://iglucentral.com/schemas/com.snowplowana
lytics.self-desc/schema/jsonschema/1-0-0#",
                "description": "JSON schema for
a button click  event",
                "self": {
                                "vendor":
"com.acme",
                                "name": "click",
                                "format":
"jsonschema",
                                "version": "1-0-
1"
                },

                "type": "object",
                "properties": {
                                "button": {
                                  "type":
["string", "null"],
                                  "maxLength":
255
                                },
                                "index": {
                                  "type":
["integer", "null"]
                                }
                },
                "additionalProperties": false
}
```

The real benefit of that approach is **how** the data then looks

Default fields (1/130)	Custom Event data
event_name	**unstruct_event_com_acme_click_1**

click	{
	"button": "open_article"
	}

Default fields (1/130)	Custom Event data
event_name	**unstruct_event_com_acme_click_1**

click	{
	"button": "open_article"
	}

click	{
	"button": "open_article",
	"index": "2"
	}

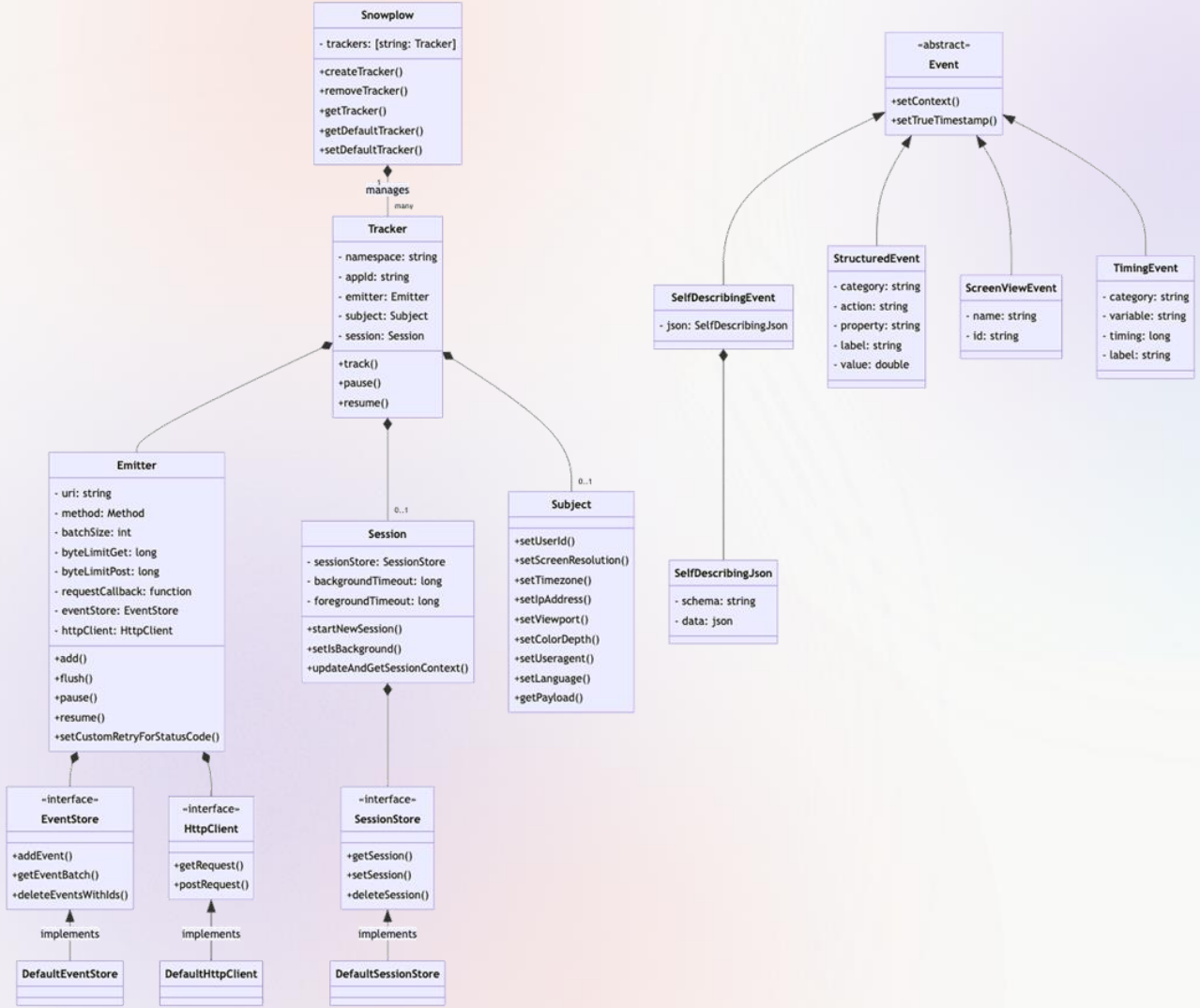
_02.1

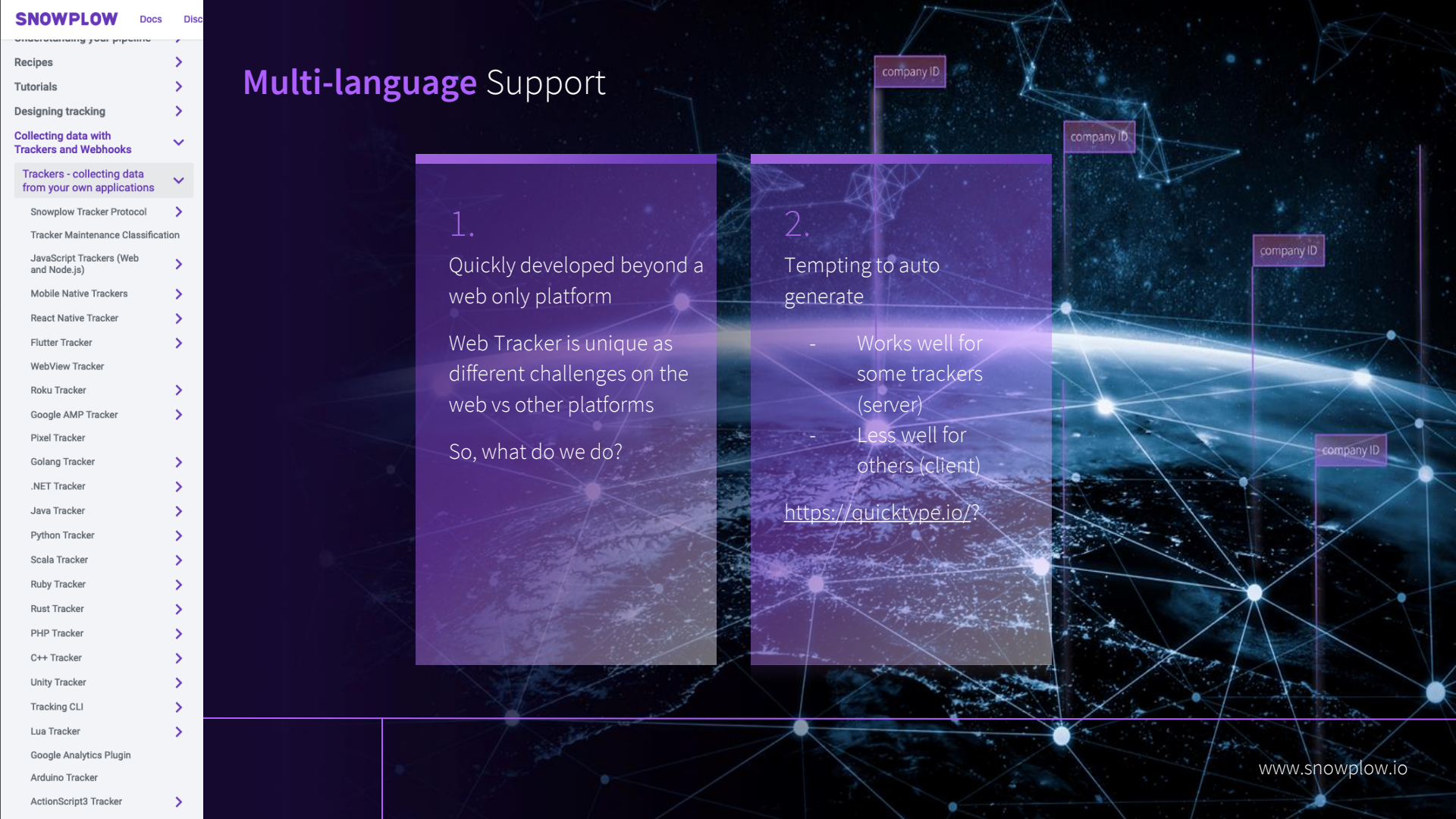
Tracking SDKs

- Lorem ipsum

Snowplow Tracking SDKs

An Introduction


Snowplow
- trackers: [string: Tracker]
+createTracker()
+removeTracker()
+getTracker()
+getDefaultTracker()
+setDefaultTracker()
1
manages
many
Tracker
- namespace: string
- appId: string
- emitter: Emitter
- subject: Subject
- session: Session
+track()
+pause()
+resume()
«abstract»
Event
+setContext()
+setTrueTimestamp()
SelfDescribingEvent
- json: SelfDescribingJson
StructuredEvent
- category: string
- action: string
- property: string
- label: string
- value: double
ScreenViewEvent
- name: string
- id: string
TimingEvent
- category: string
- variable: string
- timing: long
- label: string
Emitter
- uri: string
- method: Method
- batchSize: int
- byteLimitGet: long
- byteLimitPost: long
- requestCallback: function
- eventStore: EventStore
- httpClient: HttpClient
+add()
+flush()
+pause()
+resume()
+setCustomRetryForStatusCode()
0..1
Session
- sessionStore: SessionStore
- backgroundTimeout: long
- foregroundTimeout: long
+startNewSession()
+setIsBackground()
+updateAndGetSessionContext()
0..1
Subject
+setUserId()
+setScreenResolution()
+setTimezone()
+setIpAddress()
+setViewport()
+setColorDepth()
+setUseragent()
+setLanguage()
+getPayload()
SelfDescribingJson
- schema: string
- data: json
«interface»
EventStore
+addEvent()
+getEventBatch()
+deleteEventsWithIds()
«interface»
HttpClient
+getRequest()
+postRequest()
«interface»
SessionStore
+getSession()
+setSession()
+deleteSession()
implements
implements
implements
DefaultEventStore
DefaultHttpClient
DefaultSessionStore


SNOWPLOW
Docs
Disc
Recipes
Tutorials
Designing tracking
Collecting data with Trackers and Webhooks
Trackers - collecting data from your own applications
Snowplow Tracker Protocol
Tracker Maintenance Classification
JavaScript Trackers (Web and Node.js)
Mobile Native Trackers
React Native Tracker
Flutter Tracker
WebView Tracker
Roku Tracker
Google AMP Tracker
Pixel Tracker
Golang Tracker
.NET Tracker
Java Tracker
Python Tracker
Scala Tracker
Ruby Tracker
Rust Tracker
PHP Tracker
C++ Tracker
Unity Tracker
Tracking CLI
Lua Tracker
Google Analytics Plugin
Arduino Tracker
ActionScript3 Tracker
Multi-language Support
1.
Quickly developed beyond a web only platform
Web Tracker is unique as different challenges on the web vs other platforms
So, what do we do?
2.
Tempting to auto generate
- Works well for some trackers (server)
- Less well for others (client)
https://quicktype.io/?
company ID
company ID
company ID
company ID
www.snowplow.io

Building SDKs

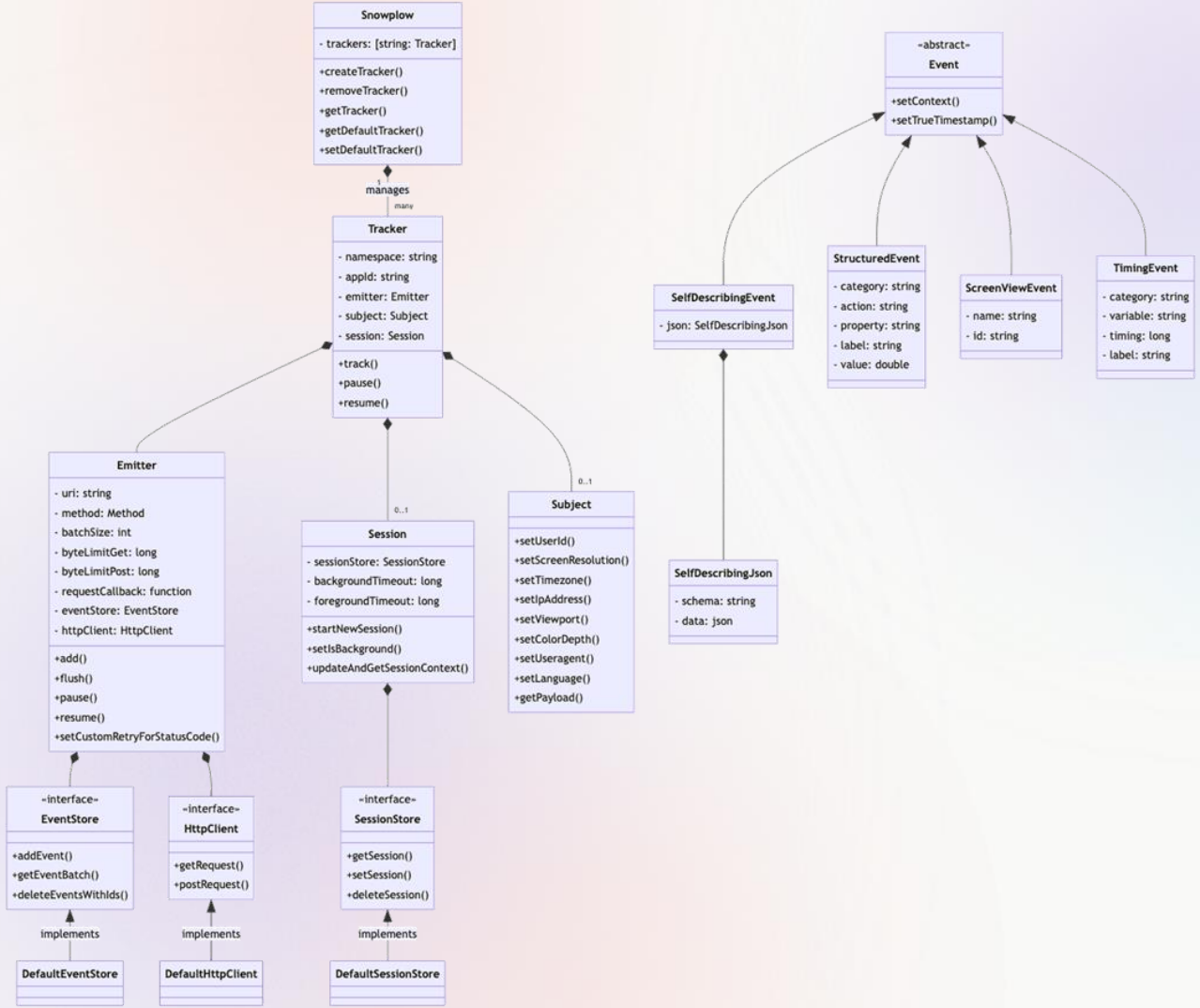
Hand craft SDKs

Autogenerate SDKs

Can we do both?

Server vs Client SDKs

How do they differ?


Snowplow
- trackers: [string: Tracker]
+createTracker()
+removeTracker()
+getTracker()
+getDefaultTracker()
+setDefaultTracker()
1
manages
many
Tracker
- namespace: string
- appId: string
- emitter: Emitter
- subject: Subject
- session: Session
+track()
+pause()
+resume()
«abstract»
Event
+setContext()
+setTrueTimestamp()
SelfDescribingEvent
- json: SelfDescribingJson
StructuredEvent
- category: string
- action: string
- property: string
- label: string
- value: double
ScreenViewEvent
- name: string
- id: string
TimingEvent
- category: string
- variable: string
- timing: long
- label: string
Emitter
- uri: string
- method: Method
- batchSize: int
- byteLimitGet: long
- byteLimitPost: long
- requestCallback: function
- eventStore: EventStore
- httpClient: HttpClient
+add()
+flush()
+pause()
+resume()
+setCustomRetryForStatusCode()
0..1
Session
- sessionStore: SessionStore
- backgroundTimeout: long
- foregroundTimeout: long
+startNewSession()
+setIsBackground()
+updateAndGetSessionContext()
0..1
Subject
+setUserId()
+setScreenResolution()
+setTimezone()
+setIpAddress()
+setViewport()
+setColorDepth()
+setUseragent()
+setLanguage()
+getPayload()
SelfDescribingJson
- schema: string
- data: json
«interface»
EventStore
+addEvent()
+getEventBatch()
+deleteEventsWithIds()
«interface»
HttpClient
+getRequest()
+postRequest()
«interface»
SessionStore
+getSession()
+setSession()
+deleteSession()
implements
implements
implements
DefaultEventStore
DefaultHttpClient
DefaultSessionStore

Configuration Challenges

```
newTracker('sp', '{{collector_url_here}}', {
  appId: 'my-app-id',
  platform: 'web',
  cookieDomain: null,
  discoverRootDomain: true,
  cookieName: '_sp_',
  cookieSameSite: 'Lax', // Recommended
  cookieSecure: true,
  encodeBase64: true,
  respectDoNotTrack: false,
  eventMethod: 'post',
  bufferSize: 1,
  maxPostBytes: 40000,
  maxGetBytes: 1000, // available in v3.4+
  postPath: '/custom/path', // Collector must be configured
  crossDomainLinker: function (linkElement) {
    return (linkElement.href === 'http://acme.de' || linkElement.id === 'crossDomainLink');
  },
  cookieLifetime: 63072000,
  stateStorageStrategy: 'cookieAndLocalStorage',
  maxLocalStorageQueueSize: 1000,
  resetActivityTrackingOnPageView: true,
  connectionTimeout: 5000,
  anonymousTracking: false,
  // anonymousTracking: { withSessionTracking: true },
  // anonymousTracking: { withSessionTracking: true, withServerAnonymisation: true },
  contexts: {
    webPage: true, // Default
    session: false // Adds client session context entity to events, off by default. Available in v3
  },
  retryStatusCodes: [],
  dontRetryStatusCodes: []
});
```

```
newTracker('sp', '{{collector_url_here}}', {
  appId: 'my-app-id',
  cookieSameSite: 'Lax', // Recommended
});
```

Sensible defaults

Whilst the SDKs are incredibly configurable, opting for sensible defaults keeps users happy

Where possible we now try to be more opinionated and don't offer configuration

Works well until someone asks us if they can configure it on Github!

_03

Data Models

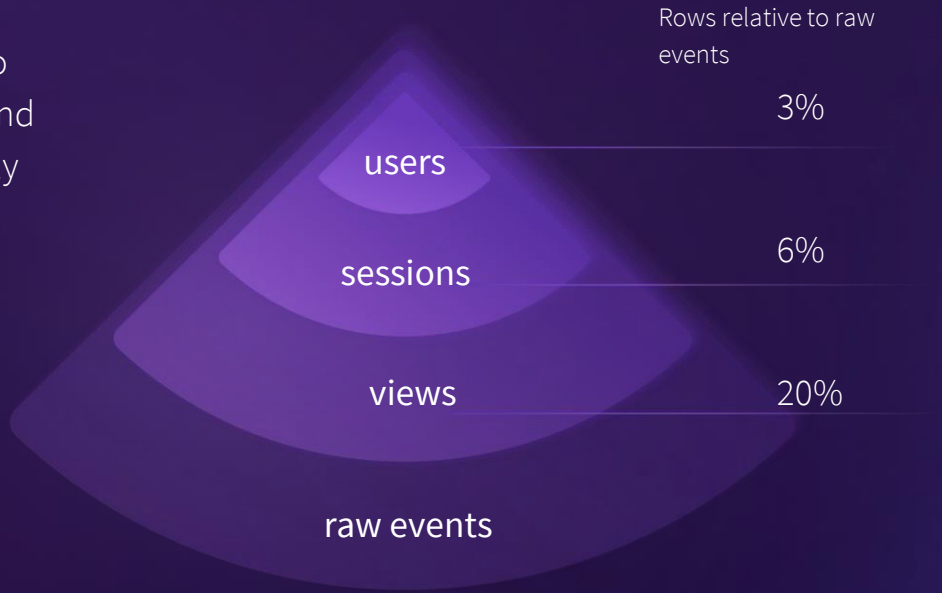
Data modeling is the process of using **business logic** to aggregate or otherwise transform **raw data**.

Modeling is not an afterthought

1. Process clickstream data from raw events to create aggregate tables of views, sessions and users, reducing the volume of data massively while adding quality

2. Deal with user stitching across sessions and devices

3. Compatible with schema evolution, so when you add new fields to your contexts they immediately show up in your data models



This end-to-end approach results in

Clean, structured data always available in your warehouse

Consistent business logic defined once means no qualms about what metrics mean

More time spent building out ML/AI models instead of toiling with data problems

Raw **Data**

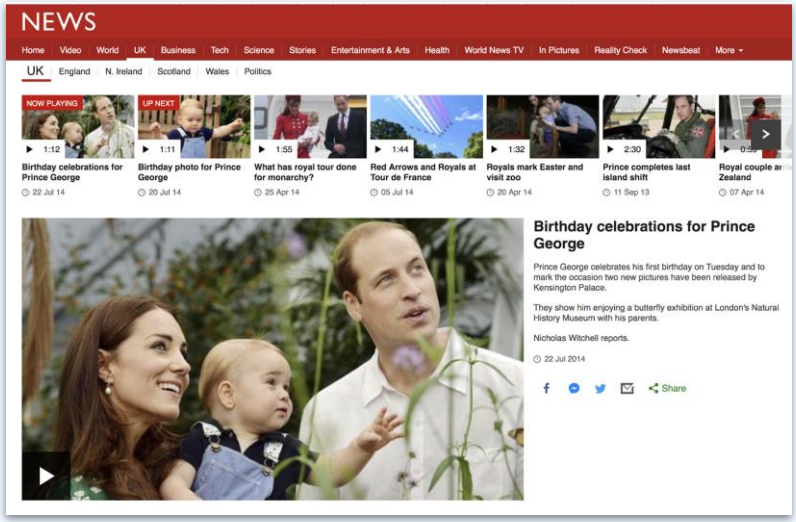
SNOWPLOW

What does the data look like

	Out-of-the-box fields (3/130)			Search event	Checkout event		Transaction event	User entity		Transaction entity	Product entity		Product entity	
	Event	**App ID**	**cookie**	**Keywords**	**Step**	**coupon**	**Revenue**	**Name**	**Type**	**ID**	**Product**	**Value**	**Product**	**Value**
1	Page view	web	a1											
2	Register	web	a1					joe	buyer					
3	Search	web	a1	wheels				joe	buyer		wheel_set	499.99	wheel	199.99
4	Link click	web	a1					joe	buyer		wheel_set	499.99		
5	Checkout	web	a1		Add basket			joe	buyer		wheel_set	499.99		
6	Email open	email	a1											
7	Checkout	iOS app			Coupon	blckfrdy		joe	buyer		wheel_set	499.99	plow	3499.99
8	Checkout	iOS app			Checkout			joe	buyer	def34	wheel_set	499.99	plow	3499.99
9	Transaction	Server					2799.99			def34	wheel_set	499.99	plow	3499.99
10	Return	Server								def34	plow	3499.99		

SNOWPLOW

Structure data to match your product



Events

Entity

content

```
id: 'abc123'
type: 'video'
date updated: 2019-06...
title: 'Birthday…'
creator: 'Nicholas Witchell'
theme: 'royals'
native: FALSE
personalities: ['Kate',
                'Wills',
                'George']
```

view	
video_play	
video_pause	
heartbeat	
click	
search	**content** **content** **content**

_03.2

Going Incremental

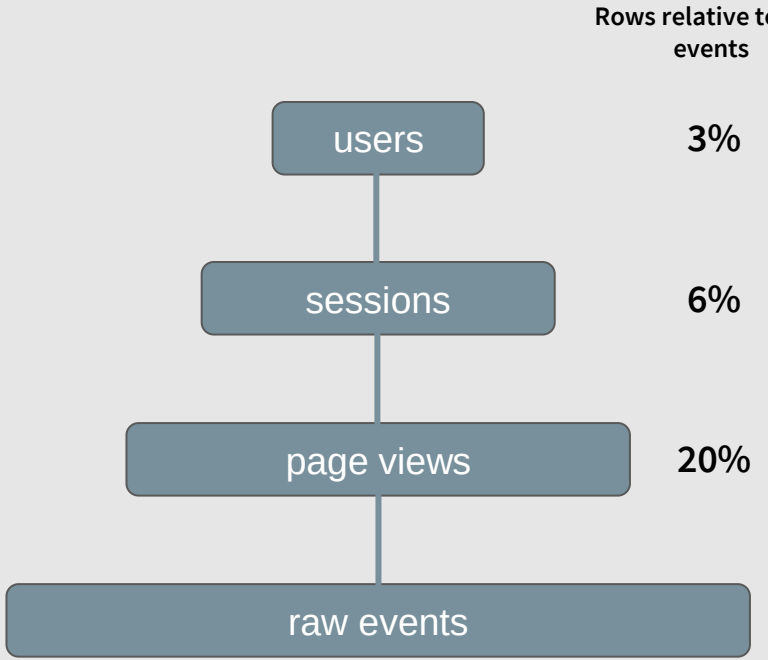
Consolidation

Similarities between queries:

- Common levels of aggregation
- Repeated logic, like joins

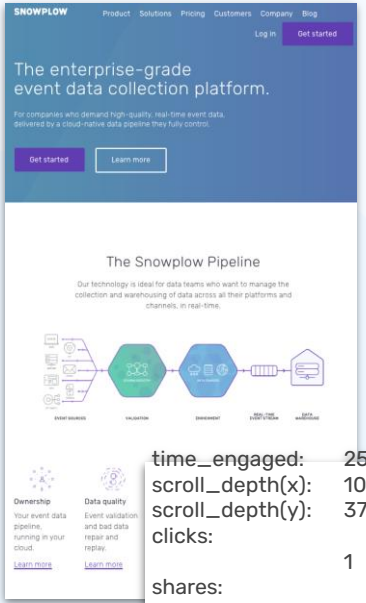
Consolidate ad-hoc queries into a set of derived tables.

These generalised tables can be used to solve a variety of use cases.

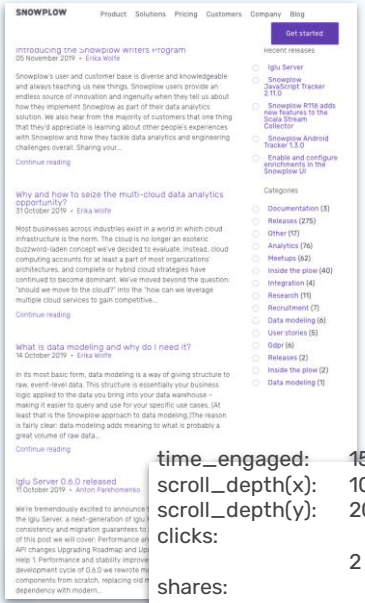


SNOWPLOW

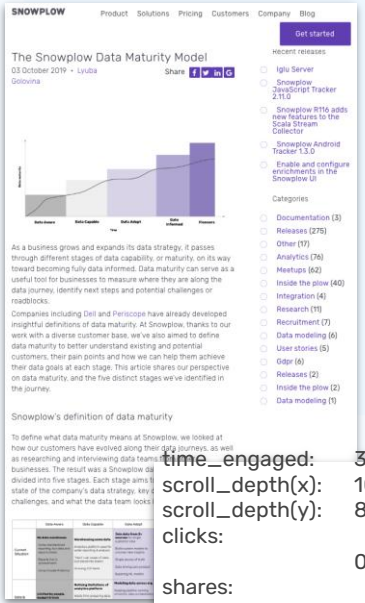
Value of a page or screen view



time_engaged:	25s
scroll_depth(x):	100%
scroll_depth(y):	37%
clicks:	1
shares:	0



time_engaged:	15s
scroll_depth(x):	100%
scroll_depth(y):	20%
clicks:	2
shares:	0



time_engaged:	35s
scroll_depth(x):	100%
scroll_depth(y):	86%
clicks:	0
shares:	1

PAGE VIEW

HEARTBEAT

ENGAGEMENT

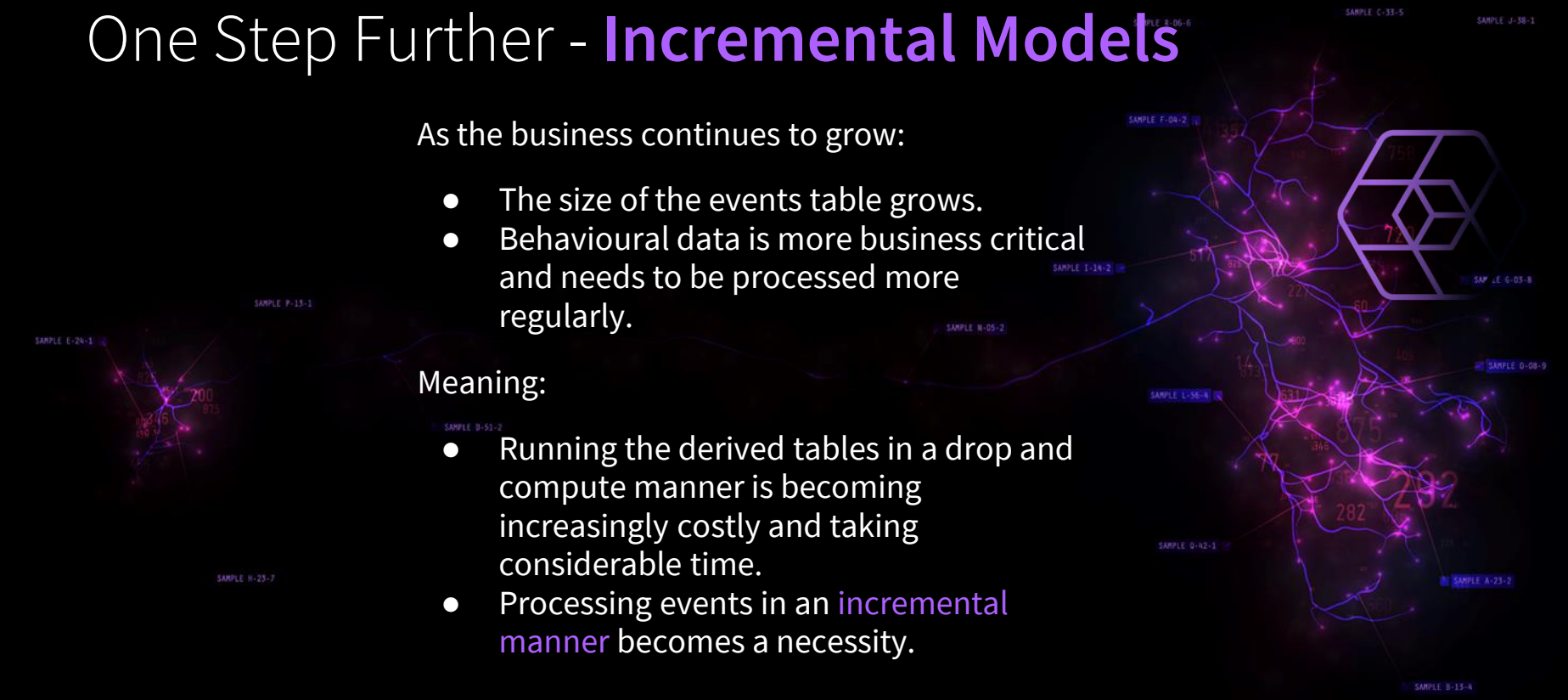
One Step Further - **Incremental Models**

As the business continues to grow:

- The size of the events table grows.
- Behavioural data is more business critical and needs to be processed more regularly.

Meaning:

- Running the derived tables in a drop and compute manner is becoming increasingly costly and taking considerable time.
- Processing events in an incremental manner becomes a necessity.



Page views - **Incremental**

Drop & Recompute

```
#page_views.sql
{{ config(
    materialized='table'
    )
}}

select
  page_view_id,
  ...
  row_number() over (
    partition by session_id
    order by derived_tstamp
    ) AS page_view_in_session_index

from {{ ref('events') }}

where event_name = 'page_view'
```

Incremental

```
#page_views.sql
{{ config(
    materialized='incremental',
    unique_key='page_view_id'
    )
}}

with sessions_with_new_events as (
  select distinct
    session_id

  from {{ ref('events') }}
  where event_name = 'page_view'
  {% if is_incremental() %}
    and derived_tstamp > (
      select max(derived_tstamp) from
{{this}}
      )
  {% endif %}
)

select
  e.page_view_id,
  ...
  row_number() over (
    partition by e.session_id
    order by e.derived_tstamp
    ) AS page_view_in_session_index

from {{ ref('events') }} e
inner join sessions_with_new_events s
on e.session_id = s.session_id
where e.event_name = 'page_view'
```

Process the least data possible

Reduce the amount of data by:

- Ensure you filter on the partition/sort keys of the source

```
#page_views.sql
{{ config(
    materialized='incremental',
    unique_key='page_view_id'
    )
}}

with sessions_with_new_events as (
  select distinct
    session_id

  from {{ ref('events') }}
  where event_name = 'page_view'
  {% if is_incremental() %}
    and derived_tstamp > (
      select max(derived_tstamp) from
{{this}}
      )
  {% endif %}
)
```

Process the least data possible

Reduce the amount of data by:

- Ensure you filter on the partition/sort keys of the source

```
#page_views.sql
{{ config(
    materialized='incremental',
    unique_key='page_view_id'
    )
}}

with sessions_with_new_events as (
  select distinct
    session_id

  from {{ ref('events') }}
  where event_name = 'page_view'
  {% if is_incremental() %}
    and collector_tstamp > (
      select max(collector_tstamp)
from {{this}}
      )
  {% endif %}
)
```

Process the least data possible

Reduce the amount of data by:

- Ensure you filter on the partition/sort keys of the source
- Restrict table scans on all source tables if possible

```
#page_views.sql
{{ config(
    materialized='incremental',
    unique_key='page_view_id'
    )
}}

with sessions_with_new_events as (
  select ...
)

select
  e.page_view_id,
  ...
  row_number() over (
    partition by e.session_id
    order by e.derived_tstamp
    ) AS page_view_in_session_index

from {{ ref('events') }} e
inner join sessions_with_new_events s
on e.session_id = s.session_id
where e.event_name = 'page_view'
```

Process the least data possible

Reduce the amount of data by:

- Ensure you filter on the partition/sort keys of the source
- Restrict table scans on all source tables if possible

```
#page_views.sql
{{ config(
    materialized='incremental',
    unique_key='page_view_id'
    )
}}

with sessions_with_new_events as (
  select ...
)

select
  e.page_view_id,
  ...


from {{ ref('events') }} e
inner join sessions_with_new_events s
on e.session_id = s.session_id
where e.event_name = 'page_view'
-- limit table scans
{% if is_incremental() %}
    and collector_tstamp > (
      select
        dateadd(
          day,
          -3,
          max(collector_tstamp))
      from {{this}}
      )
  {% endif %}
```

Process the least data possible

Reduce the amount of data by:

- Ensure you filter on the partition/sort keys of the source
- Restrict table scans on all source tables if possible
- Understanding your warehouse

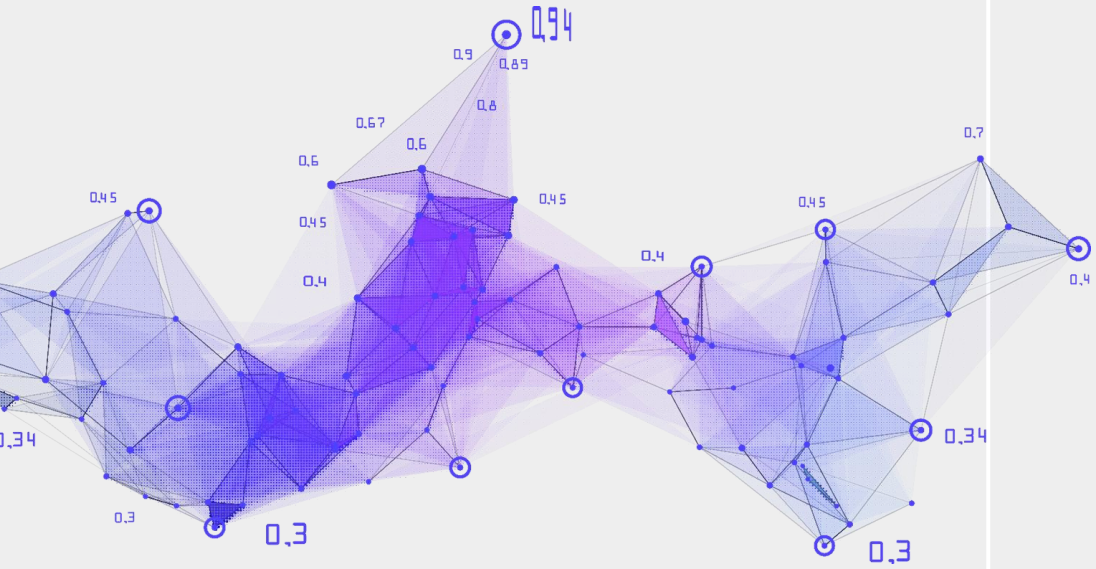
Incremental

```
#page_views.sql
{{ config(
    materialized='incremental',
    unique_key='page_view_id'
    )
}}

with sessions_with_new_events as (
  select distinct
    session_id

  from {{ ref('events') }}
  where event_name = 'page_view'
  {% if is_incremental() %}
    and derived_tstamp > (
      select max(derived_tstamp) from
{{this}}
      )
  {% endif %}
)
```

Wrap up



Considered the Snowplow Tracking SDKs

- Why we hand craft them
- Differences in Server vs Client SDKs

How do we model billions of rows of atomic data?

- Incrementally!
- Aggregation brings many benefits for analysis

Configuration is painful for everyone

- Easy to get carried away with configurable trackers but then Data Models need to support it too

Schema'd events make it easy to make type safe objects for us with the Tracking SDKs

- Great for tracking engineers

Understand your full pipeline to extract the most

- How the data is tracked and processed impacts how you can model the data