First Friday Tech Happy Hour

EVENT STORE®

Axian CREATIVITY, SOFTWARE, BUSINESS SOLUTIONS

EventStoreDB

The database for Event Sourcing

HISTORY

Greg Young

2007 - Formalized CQRS/ES
2012 - Released EventStore v1

"When you start modelling events, it forces you to think about the behaviour of the system. As opposed to thinking about the structure of the system."

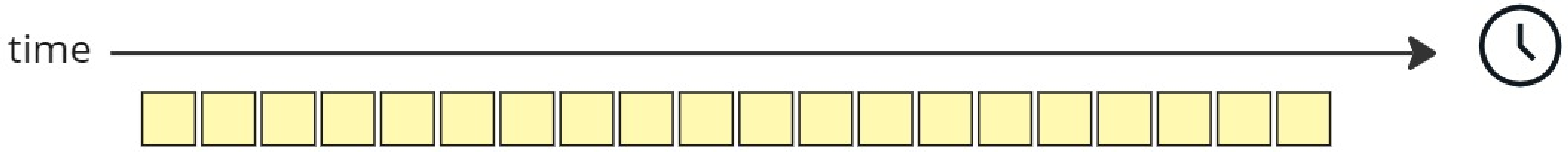
Meet Ouro!

The EventStore Mascot

Axian CREATIVITY, SOFTWARE, BUSINESS SOLUTIONS

What's EventStoreDB all about?

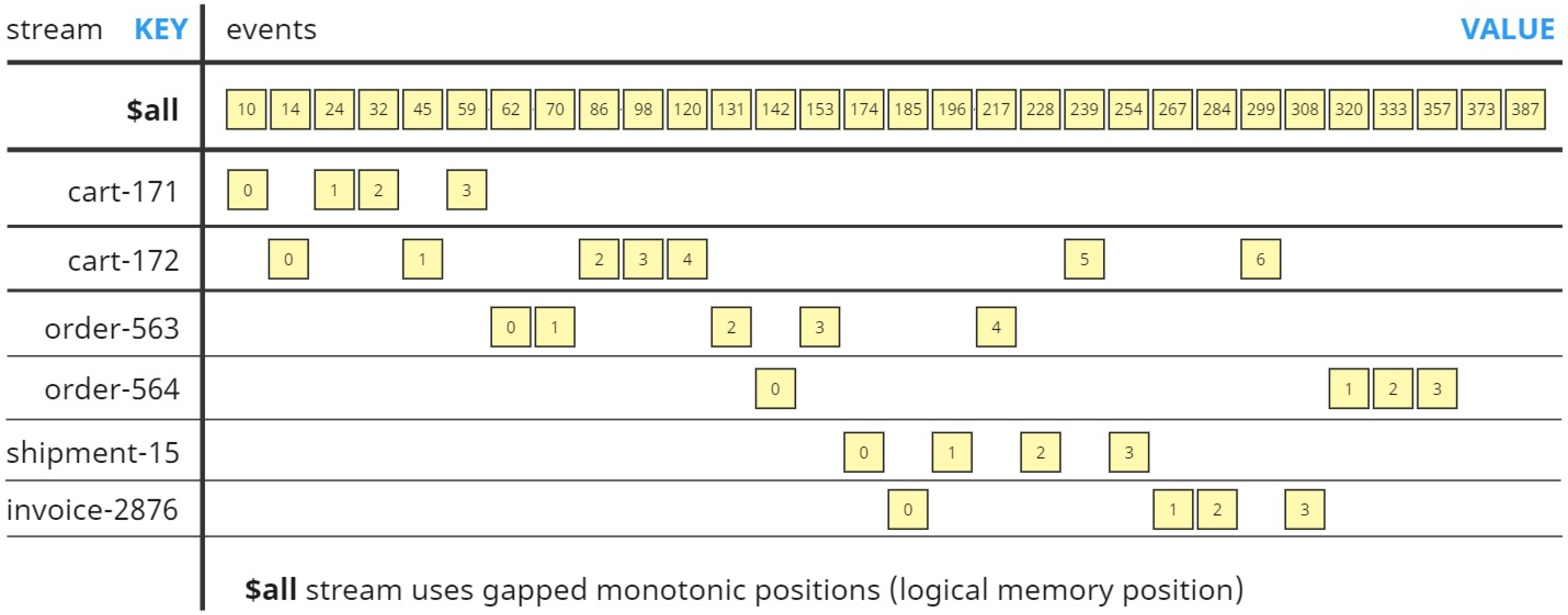
Storage of data into streams of immutable events



Guaranteed writes
Guaranteed ordering
Optimistic concurrency model
Granular streams
Flexibility in system evolution

Eventual Consistency
CQRS
No Data Loss
Key-Value Database

EventStoreDB - Logical Structure

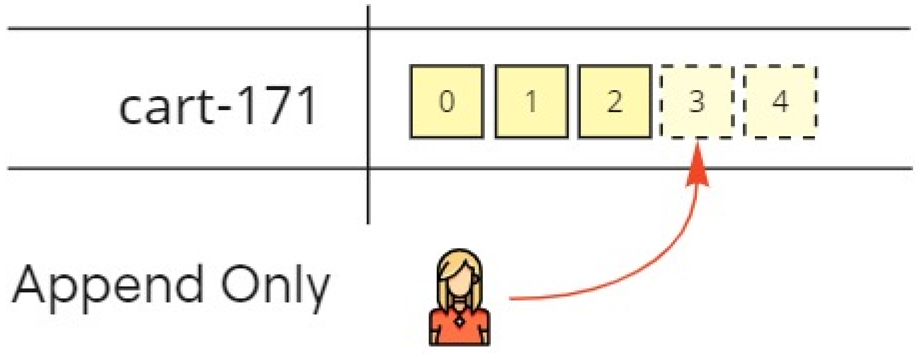


stream **KEY**	events																													**VALUE**
$all	10	14	24	32	45	59	62	70	86	98	120	131	142	153	174	185	196	217	228	239	254	267	284	299	308	320	333	357	373	387
cart-171	0		1	2		3																								
cart-172		0			1				2	3	4									5				6						
order-563							0	1				2		3				4												
order-564													0													1	2	3		
shipment-15															0		1		2		3									
invoice-2876																0						1	2		3					

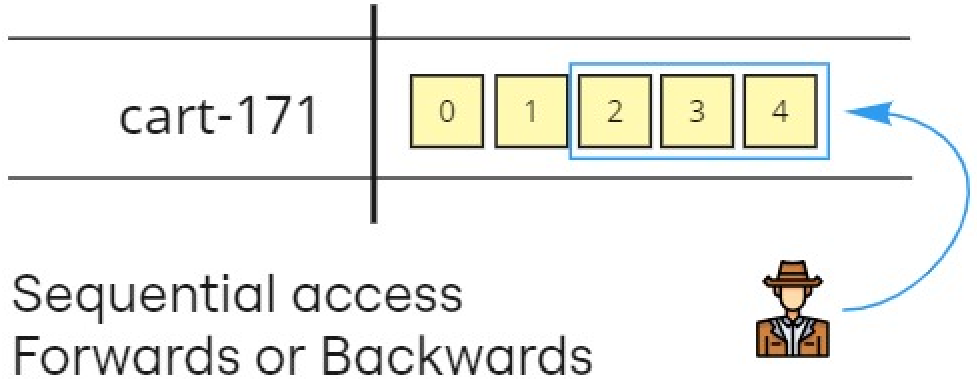
$all stream uses gapped monotonic positions (logical memory position)
Other streams use gapless monotonic stream revisions (event number)

What can you do with EventStoreDB?

Append Events to a Stream



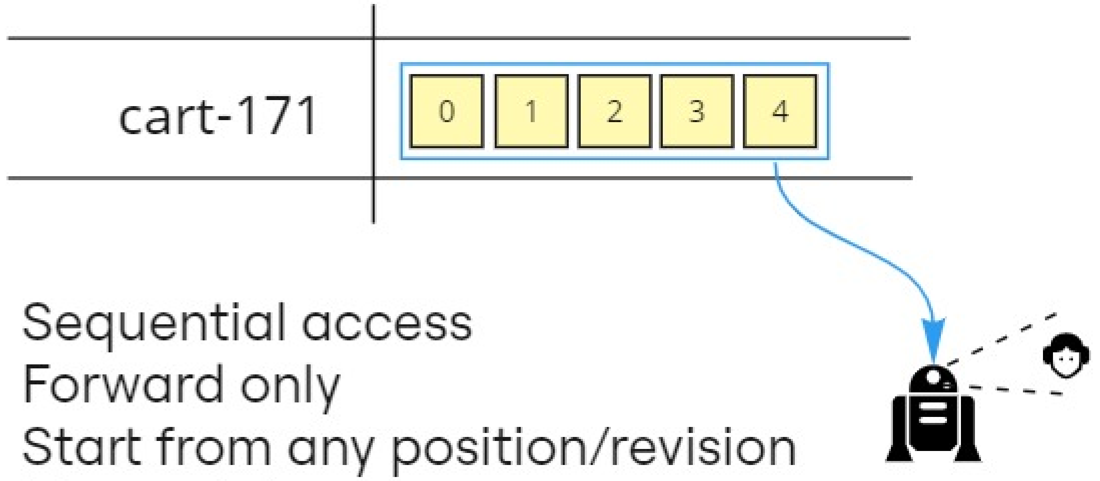
Read Events from a Stream



Sequential access
Forwards or Backwards
Start from any position/revision
Read 1 to N events

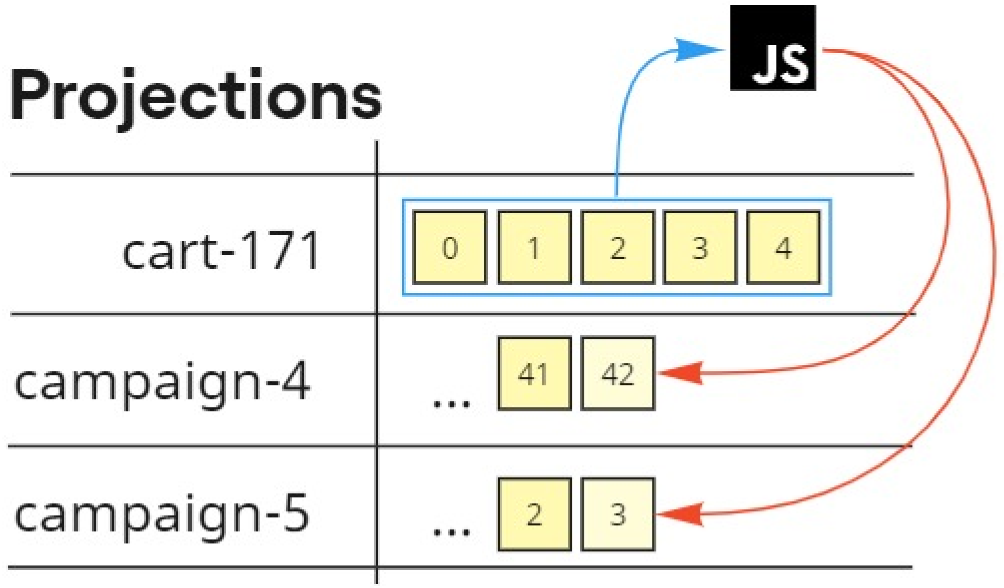
What else can you do with EventStoreDB?

Subscribe to a stream



Sequential access
Forward only
Start from any position/revision
Live updates
Catch-up Subscriptions
Persistent Subscriptions

Projections



Append new events or link existing events to streams

Temporal correlation queries
Built in system projections
User defined JavaScript projections

⚠ Write Amplification increases I/O load on Leader

! Projections and Applications cannot append events to the same stream.

Append Events to a Stream

EventData (before persist)

```
{
    "id": "6a339777-3c26-4c74-8655-83811c765b11",
    "type": "gameStarted",
    "data": "{\"gameId\":1002,\"players\":[\"Jake\",\"Eric\"]}",
    "metadata": "{\"createdAt\":\"2023-09-26T06:52:55.2432897Z\"}"
}
```

Optional Application defined identifiers

Tip: Include your own creation date in event metadata (Creation Time != Persisted Time)

Use simple domain specific names for event types, not a technical detail like the .NET Fully Qualified Type Name which simplifies deserialization, at the cost of tightly coupling the domain code to the data.

EventStore DB includes the following on append:

- Adds created timestamp (persisted at timestamp)
- Generates an eventId if not specified
- Adds a stream sequence number
- Adds the global stream position ($all)

ResolvedEvent (after persist)

```
{
        "event": {
            "contentType": "application/json",
            "created": "2023-09-26T06:52:55.3765804Z",
            "data": "{\"gameId\":1002,\"players\":[\"Jake\",\"Eric\"]}",
            "eventId": "6a339777-3c26-4c74-8655-83811c765b11",
            "eventNumber": 0,
            "eventType": "gameStarted",
            "eventStreamId": "game-3",
            "metadata": "{\"createdAt\":\"2023-09-26T06:52:55.2432897Z\"}",
            "position": "C:13051/P:13051"
        },
        "link": {},
        "originalPosition": "C:13051/P:13051"
    },
```

Read Events from a Stream

Direction	Forwards or Backwards
Revision	The 0-based integer of where to start the read operation
Count	The number of events to read from the stream
ResolveLinks	Retrieve the event referenced by a link event
Scope	This is implement as a helper by the WebAPI to scope the response data to what you need.

scope = resolved

```
[{
 "event": {
  "contentType": "application/json",
  "created": "2023-09-26T06:52:55.3765804Z",
  "data": "{\"gameId\":1002,\"players\":[\"Jake\",\"Eric\"]}",
  "eventId": "6a339777-3c26-4c74-8655-83811c765b11",
  "eventNumber": 0,
  "eventType": "gameStarted",
  "eventStreamId": "game-3",
  "metadata": "{\"ec\":\"2023-09-26T06:52:55.3743212Z\"}",
  "position": "C:13051/P:13051"
 },
 "link": {
  "contentType": "application/octet-stream",
  "created": "2023-09-26T06:52:55.393416Z",
  "data": "0@game-3",
  "eventId": "31839992-90b6-4c13-ab4b-e21deca0e831",
  "eventNumber": 2,
  "eventType": "$>",
  "eventStreamId": "$streams",
  "metadata": "{\"$v\":\"1:-1:1:4\",\"$c\":13051,\"$p\":13051,\"$causedBy\":\"6a339777-3c26-4c74-8655-83811c765b11\"}",
  "position": "C:13818/P:13818"
 },
 "originalPosition": "C:13818/P:13818"
}, ...]
```

Axian CREATIVITY, SOFTWARE, BUSINESS SOLUTIONS

scope = event

```
[{
  "contentType": "application/json",
  "created": "2023-09-26T06:52:55.3765804Z",
  "data": "{\"gameId\":1002,\"players\":[\"Jake\",\"Eric\"]}",
  "eventId": "6a339777-3c26-4c74-8655-83811c765b11",
  "eventNumber": 0,
  "eventType": "gameStarted",
  "eventStreamId": "game-3",
  "metadata": "{\"ec\":\"2023-09-26T06:52:55.3743212Z\"}",
  "position": "C:13051/P:13051"
 }, ...]
```

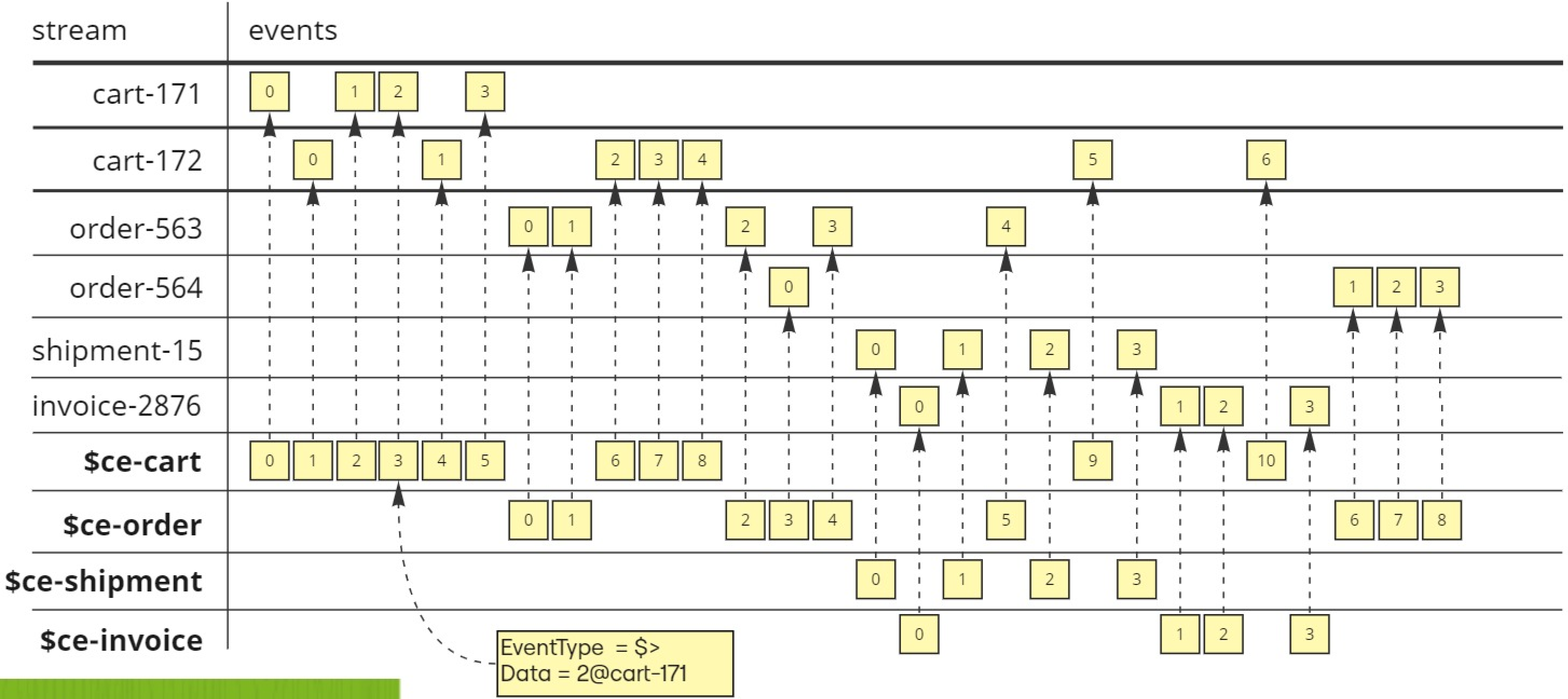
scope = data

```
[{
  "gameId":1002,
  "players":["Jake","Eric"]
 }, ...]
```

scope = metadata

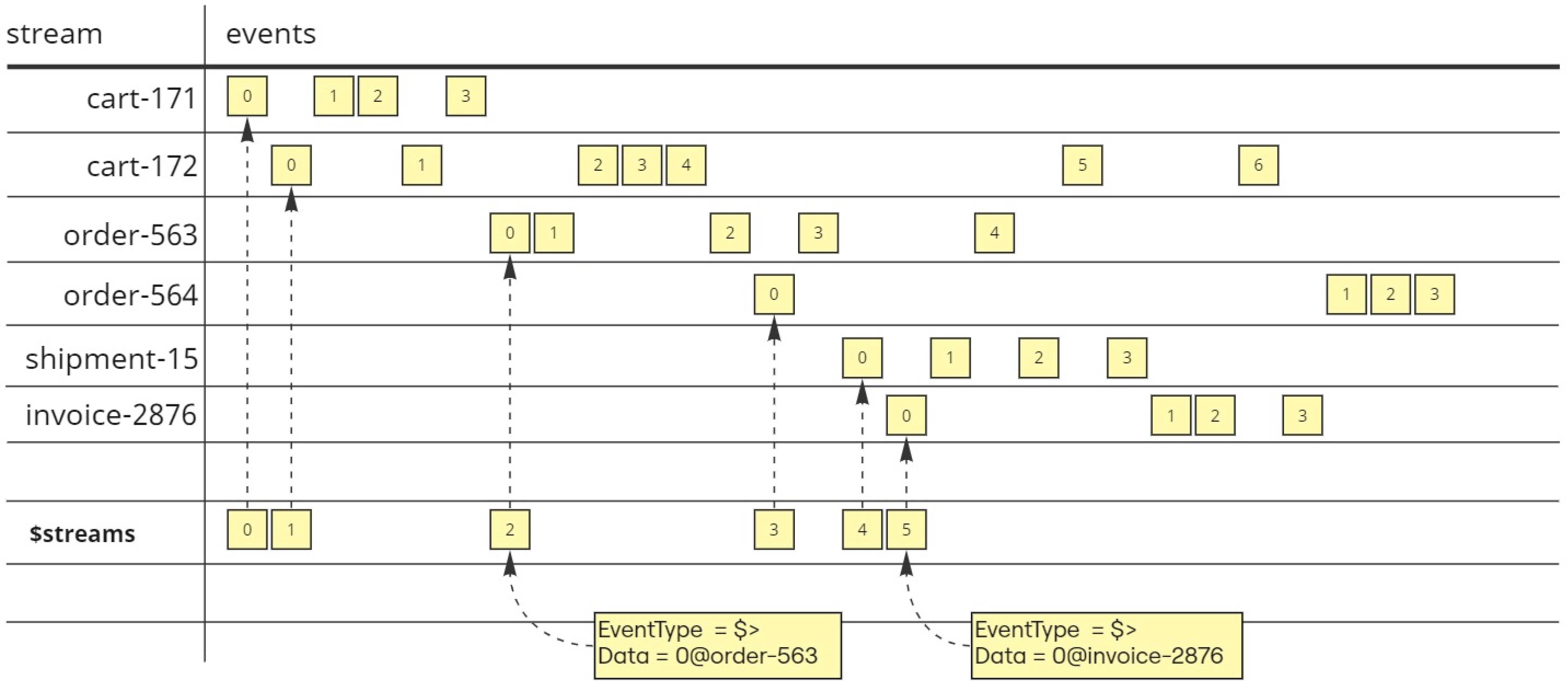
```
[{
  "ec":"2023-09-26T06:52:55.3743212Z"
 }, ...]
```

Projection: By Category

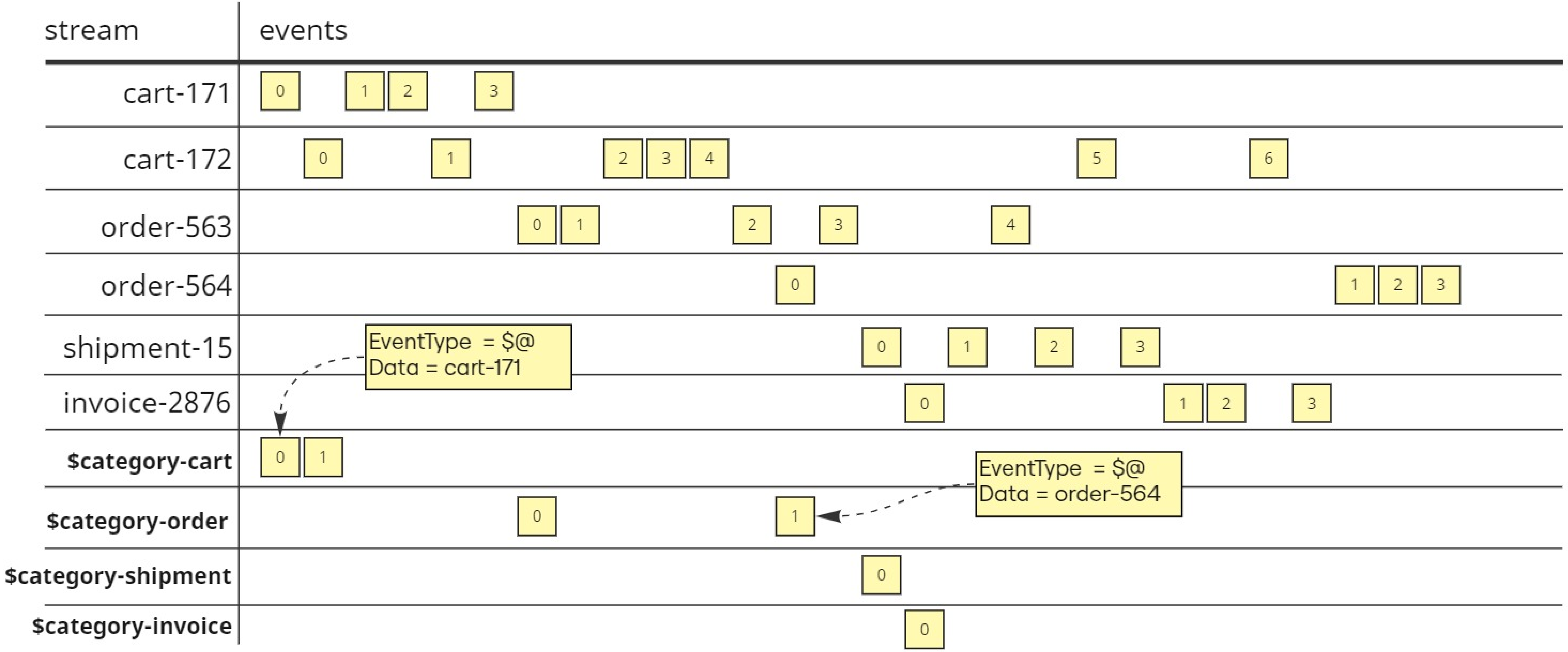


Axian CREATIVITY, SOFTWARE, BUSINESS SOLUTIONS

Projection: Streams

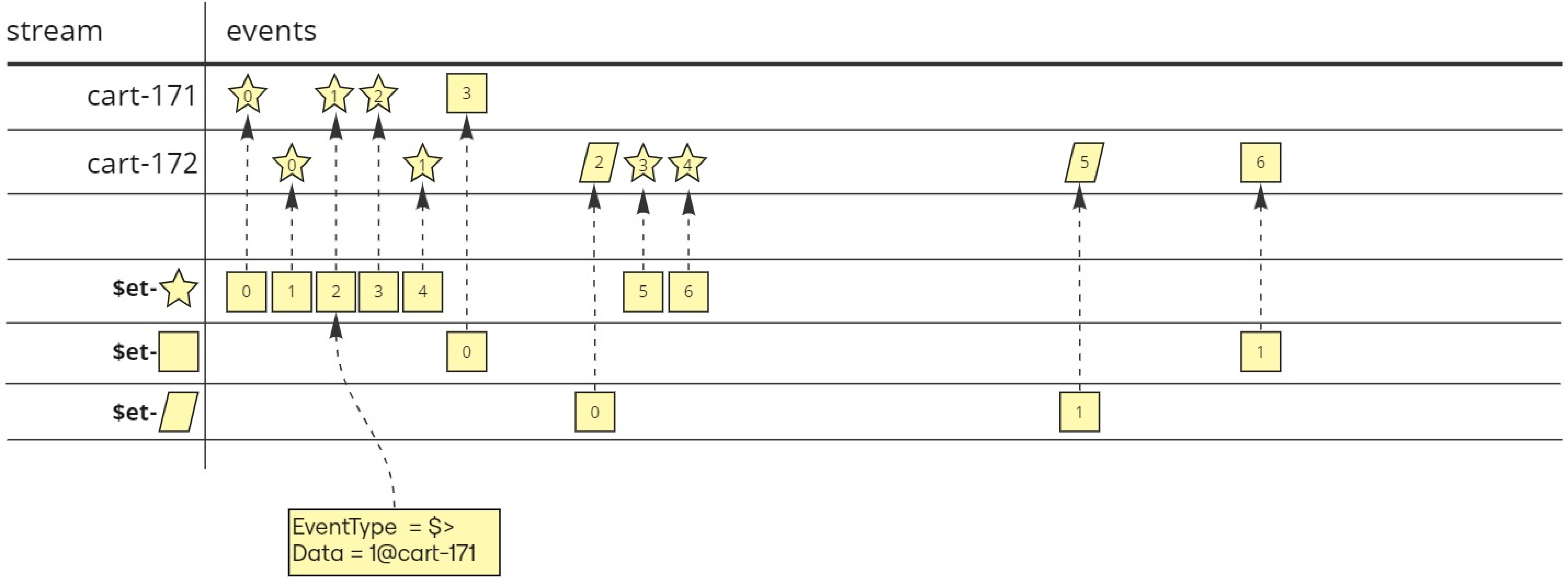


Projection: Streams By Category



Axian CREATIVITY, SOFTWARE, BUSINESS SOLUTIONS

Projection: By Event Type



EventType = $>
Data = 1@cart-171

CREATIVITY, SOFTWARE, BUSINESS SOLUTIONS