THE JSON EVOLUTION WITH SQL DATABASE

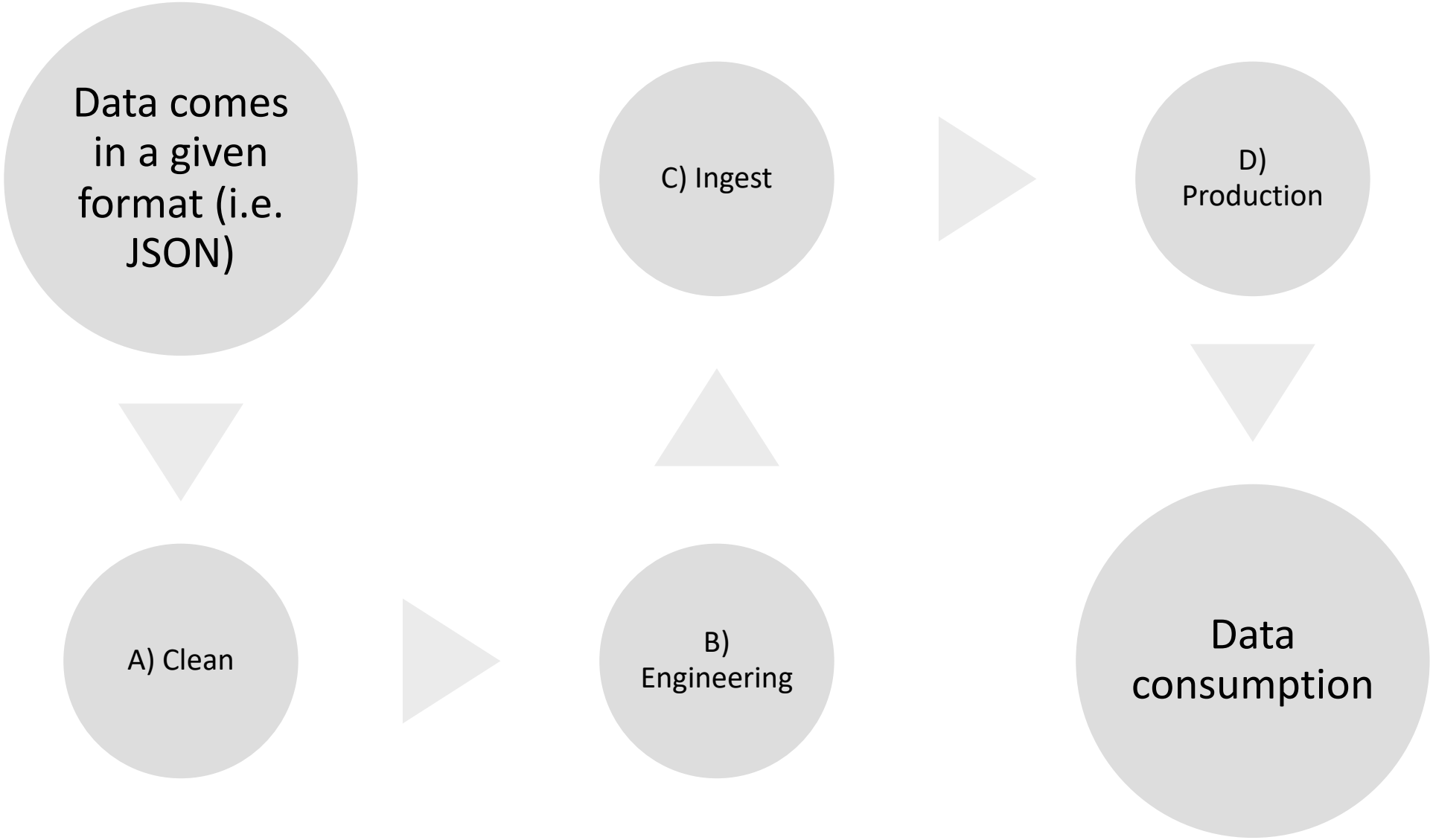
Roberto Freato – CTO @Witailer

Not a contest

Relational is great

NoSQL is awesome

This (integration) is the way


Data comes in a given format (i.e. JSON)
C) Ingest
D) Production
A) Clean
B) Engineering
Data consumption

Incoming data may change

Unexpected entities

New fields

Missing fields

Cleaning/filtering processes must be aligned

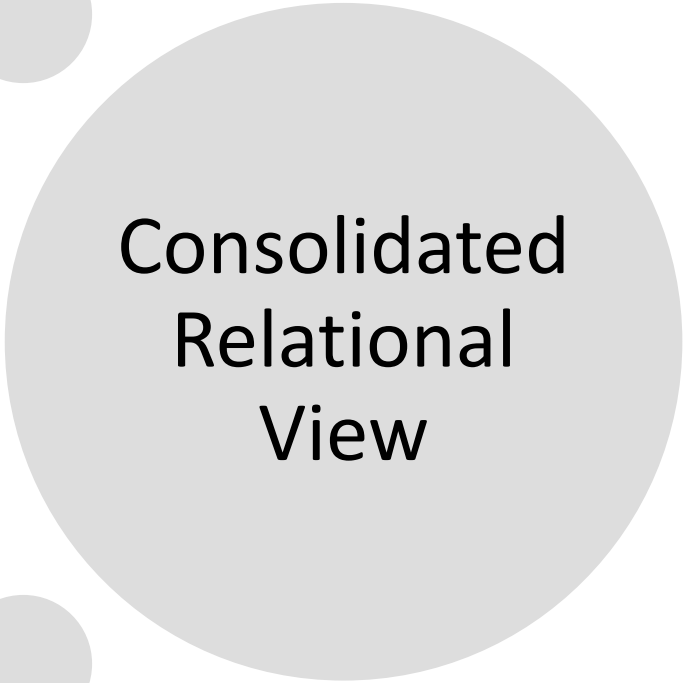
Relational tables must be enhanced

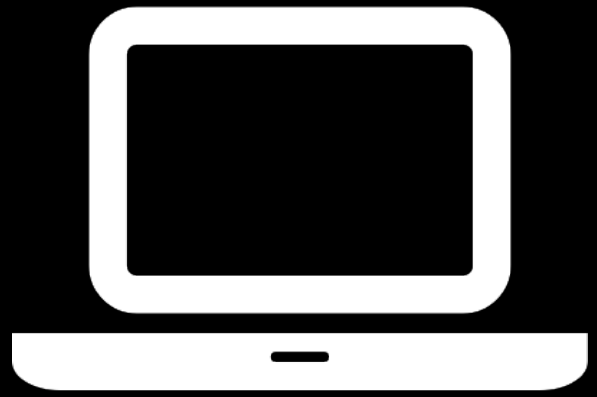
Could lead to re-indexing

Surrogated values

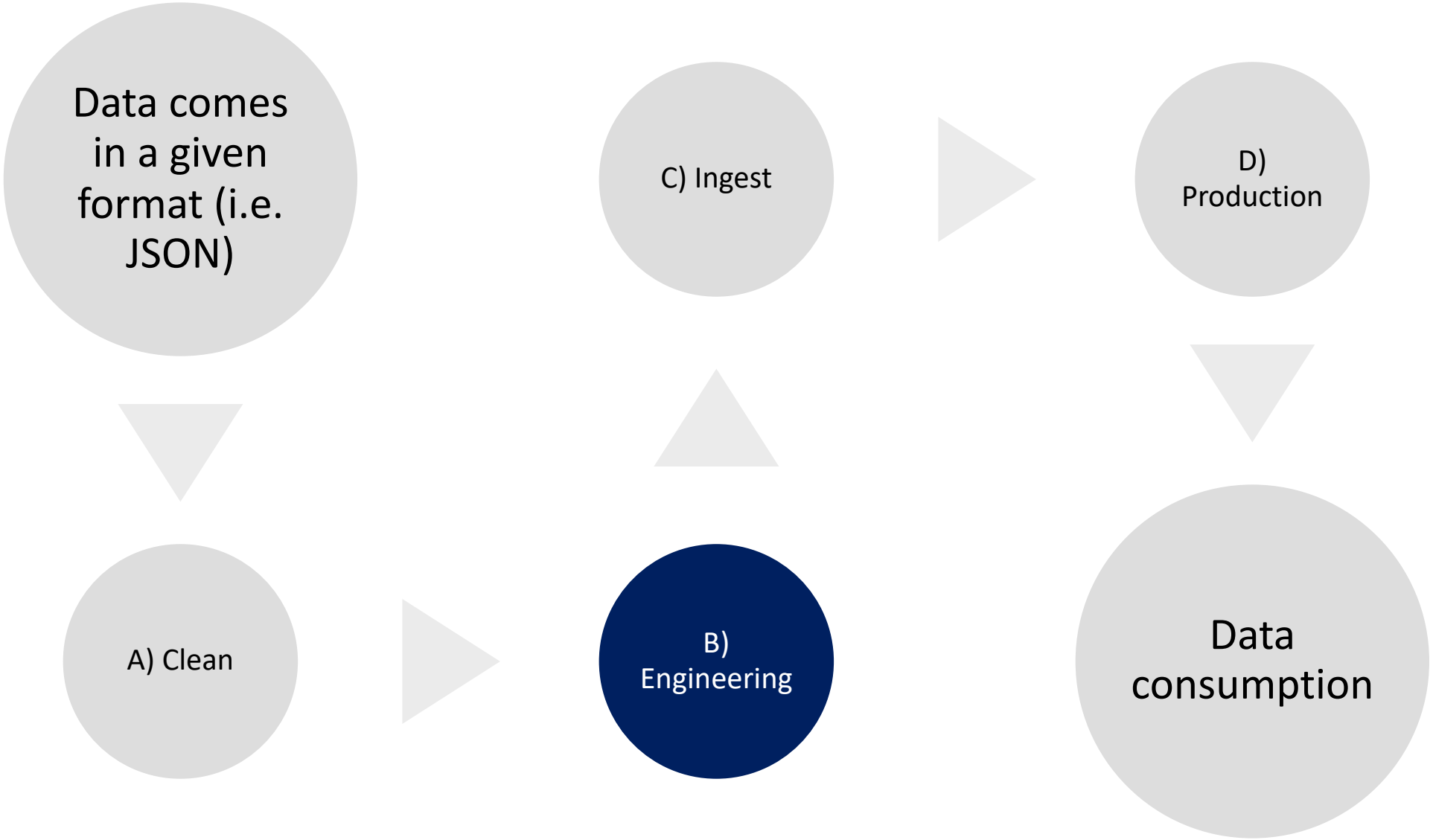
Movies JSON

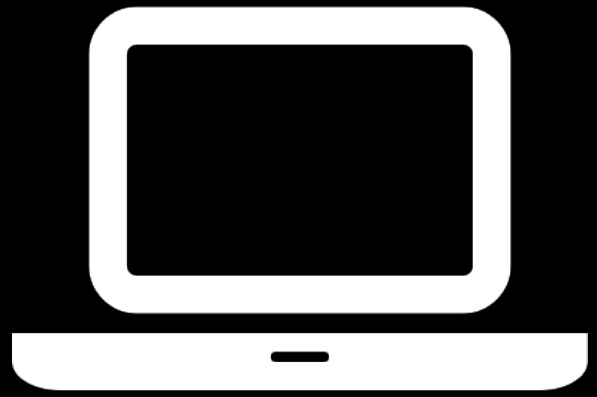
Reviews-of-movies JSON



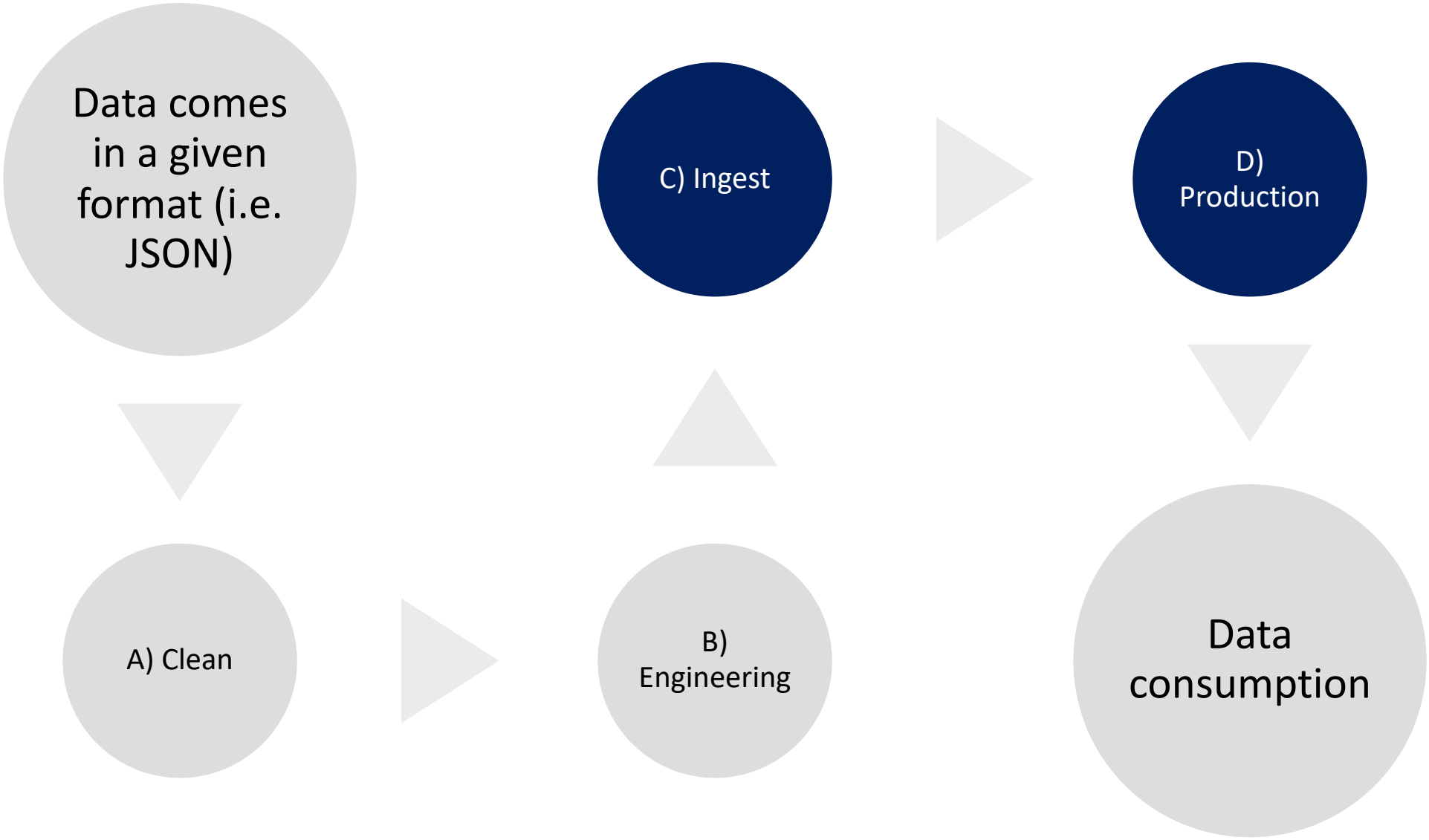


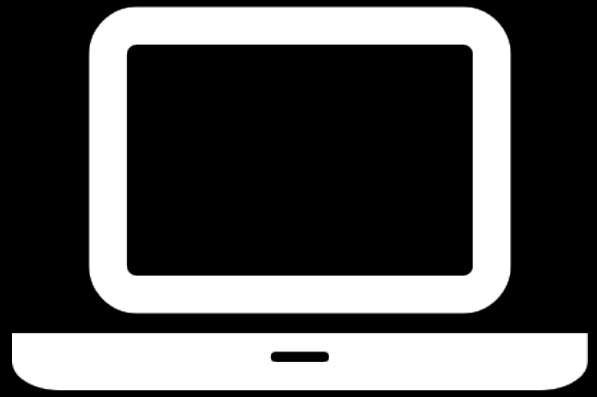
get the data


Data comes in a given format (i.e. JSON)
C) Ingest
D) Production
A) Clean
B) Engineering
Data consumption



database engineering


Data comes in a given format (i.e. JSON)
C) Ingest
D) Production
A) Clean
B) Engineering
Data consumption



ingestion and production

Keep the tables

need to edit/change the individual rows

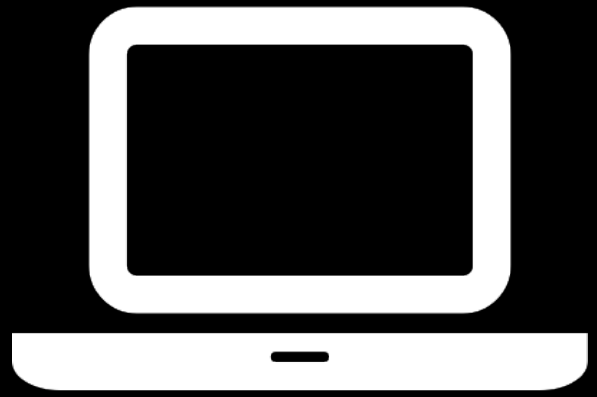
need to perform complex aggregations (avg, sum, etc)

Avoid the tables

need to produce the feed for consumption only

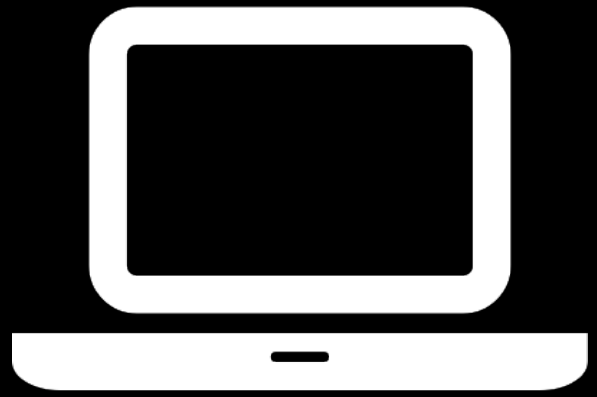
data arrives immutable

incoming data changes or it is not 100% reliable in structure



hybrid and full-json approach

ISJSON	Just checks the format
JSON_VALUE	Takes the scalar value
JSON_QUERY	Takes the object or array value
OPENJSON	Transform arrays into relational rows

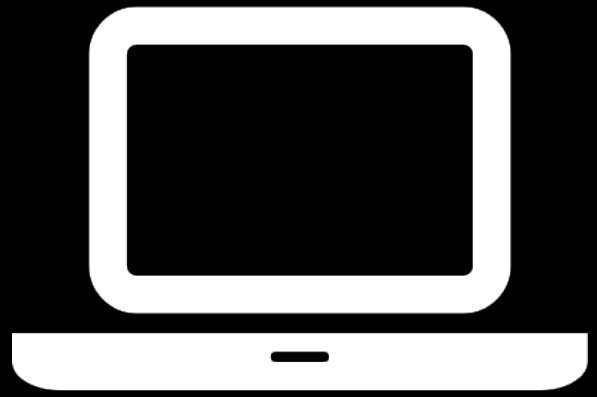


some fun

Data does not change anymore

JSON becomes huge

Need to perform analysis



denormalization

Relational stays for a huge set of needs

JSON can enhance productivity and change

Denormalization for long-term storage of JSON

JSON-in-SQL results can be improved with indexes

EFCore & JSON

PostgreSQL implementation already supports that:

(for SQL Server/Database it’s coming with 6.0)

```
public class SomeEntity
{
    public int Id { get; set; }
    [Column(TypeName = "jsonb")]
    public Customer Customer { get; set; }
}
```

Takeaways

We can store JSON into text columns in SQL Server

We can focus on views and aggregate transformation instead of making ETL at the source

We can even store huge documents in a single cell (but it’s not recommended)

In case of data growth, we can move data into real tables

And eventually make them columnar, with huge compression

With the arrival of EFCore 6.0, we can project the entities without the need of views

Thank you

Any questions?

 childotg

 childotg

 robertofreato