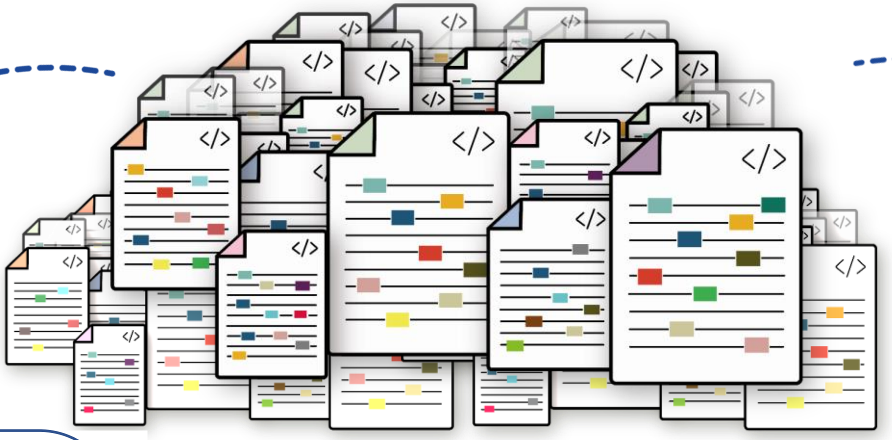
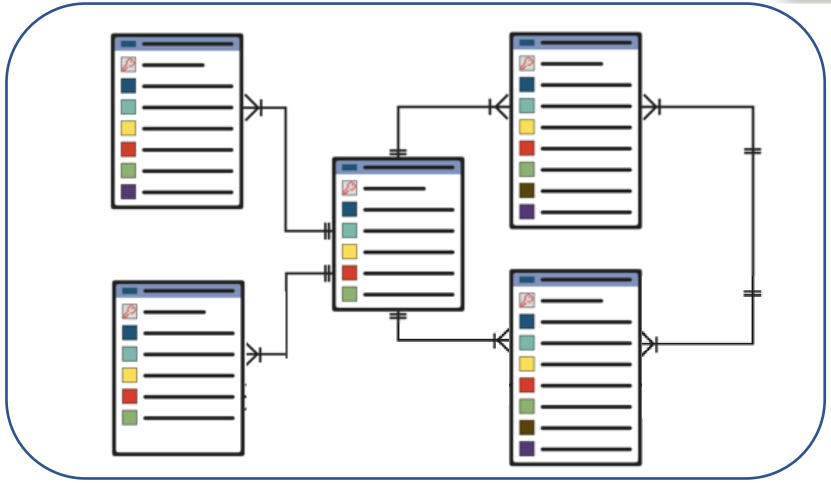
XMLParser

Imagine that you have millions of XML files

And you want to extract specific data from it and load into SQL or NoSQL database

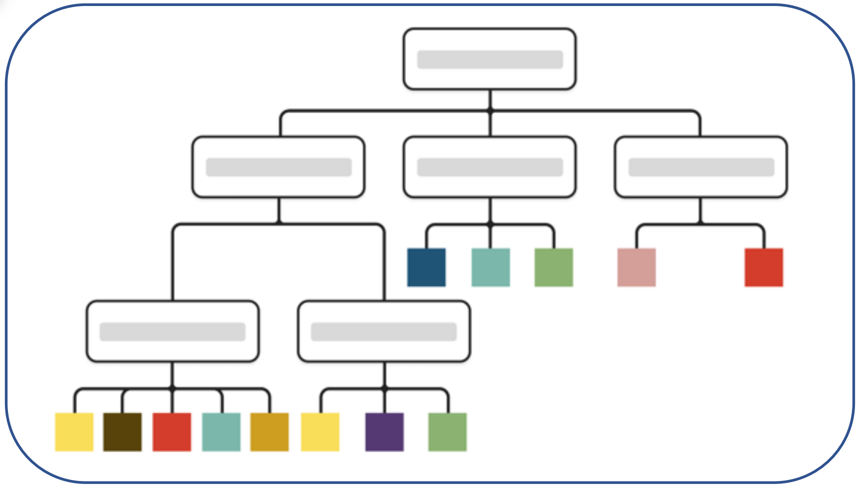


SQL Database



OR

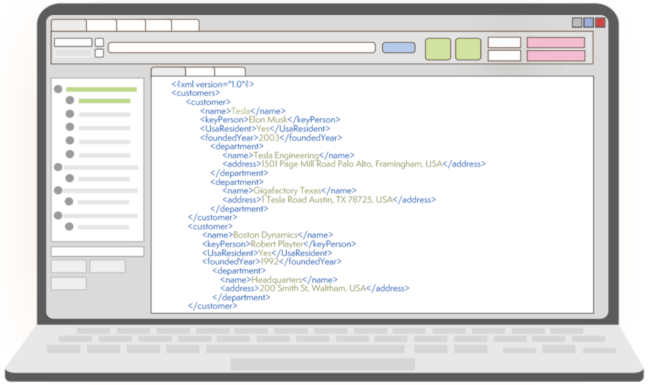
NoSQL Database



XMLParser

Our solution for you if:

- You want to have universal solution, but not single script for each document type like single developer or GPT chat most likely can do?
- You have too many file types or they have no fixed structure?
- You need very precise control over the data you extract? For example, you need extract only 15% of data? And be able to extract zip code from string “Amsterdam 1019” and place it in one column and city name in another?
- You don’t want to write code at all?



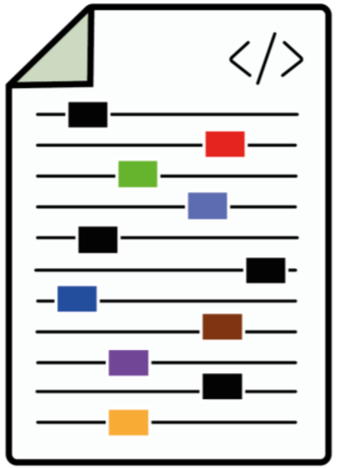
```xml
<?xml version="1.0"?>
<customers>
    <customer>
        <name>Tesla</name>
        <keyPerson>Elon Musk</keyPerson>
        <UsaResident>Yes</UsaResident>
        <foundedYear>2003</foundedYear>
           <department>
              <name>Tesla Engineering</name>
              <address>1501 Page Mill Road Palo Alto, Framingham, USA</address>
           </department>
           <department>
              <name>Gigafactory Texas</name>
              <address>1 Tesla Road Austin, TX 78725, USA</address>
           </department>
    </customer>
    <customer>
        <name>Boston Dynamics</name>
        <keyPerson>Robert Playter</keyPerson>
        <UsaResident>Yes</UsaResident>
        <foundedYear>1992</foundedYear>
           <department>
              <name>Headquarters</name>
              <address>200 Smith St, Waltham, USA</address>
           </department>
    </customer>
```

XMLParser

We provide solution that can:

Detect Morphology of Document

Extract an arbitrary set of fields



🟥 order number: 7217382

🟩 customer: John Doe

🟦 city: Amsterdam

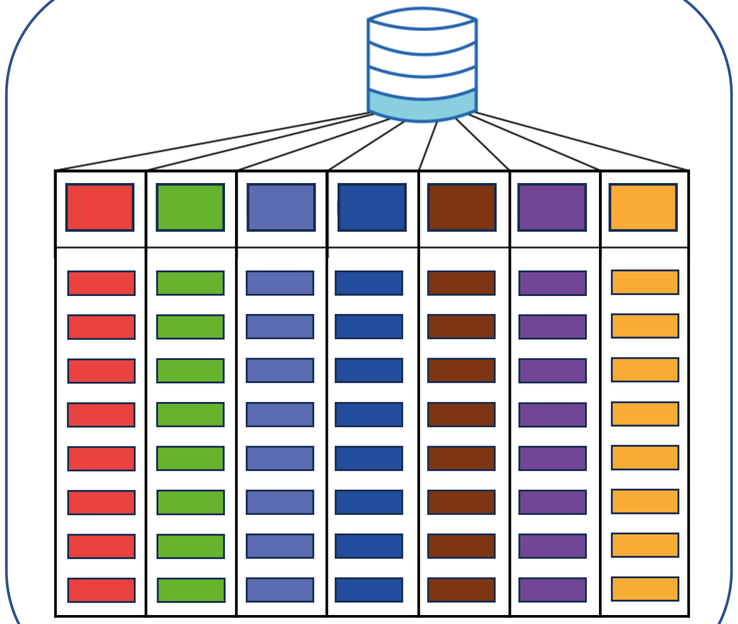
🟦 price: $1700

🟫 product name: Notebook

🟪 quantity: 1

🟧 prepared: true

Load result to database



XMLParser

1. You can define an arbitrary number of document types
2. For each type you can specify arbitrary parsing and extraction rules
3. You can get update for database records from separate documents

XMLParser

Can process documents with missing sections or with identical sections with different spelling but the same content

Expected structure:

```
<lots>
  <lot>
      <name></number>
      <price></price>
  </lot>
</lots>
```

Data with missed sections:

```
<lot>
    <name>Laptop</number>
    <price>2700</price>
</lot>
```

Restored document structure:

```
<lots>
  <lot>
    <object>
      <name>Laptop</number>
      <price>2700</price>
    </object>
  </lot>
</lots>
```

XMLParser

Can process documents with missing sections or with identical sections with different spelling but the same content

Expected structure:

```
<lots>
  <lot>
      <name></number>
      <price></price>
  </lot>
</lots>
```

Data with unexpected naming and structure:

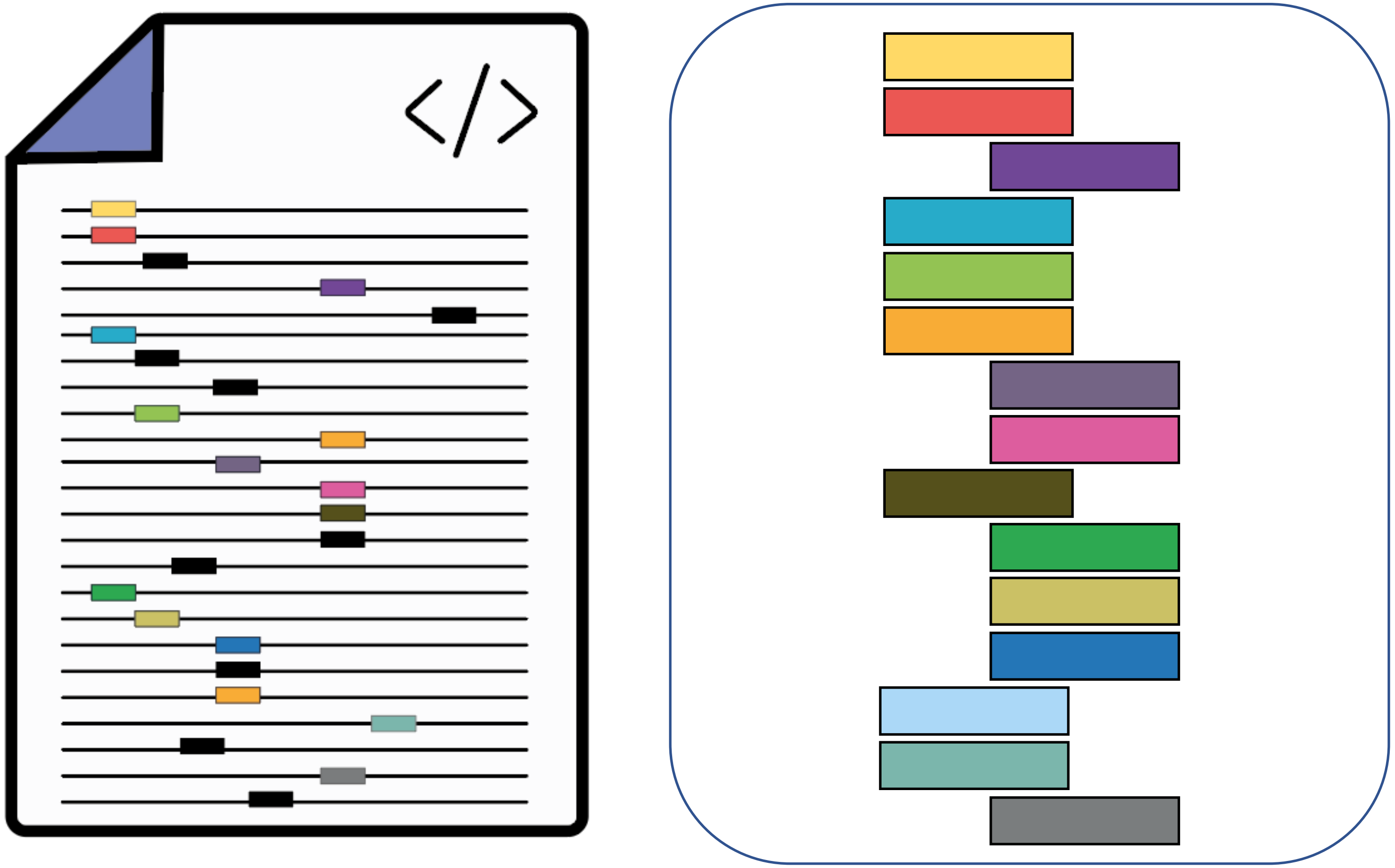
```
<lots>
  <lot-item>
    <object>
      <purchase>
       <name>Laptop</number>
       <price>2700</price>
      </purchase>
    </object>
  </lot-item>
</lots>
```

Restored document structure:

```
<lots>
  <lot>
    <object>
      <name>Laptop</number>
      <price>2700</price>
    </object>
  </lot>
</lots>
```

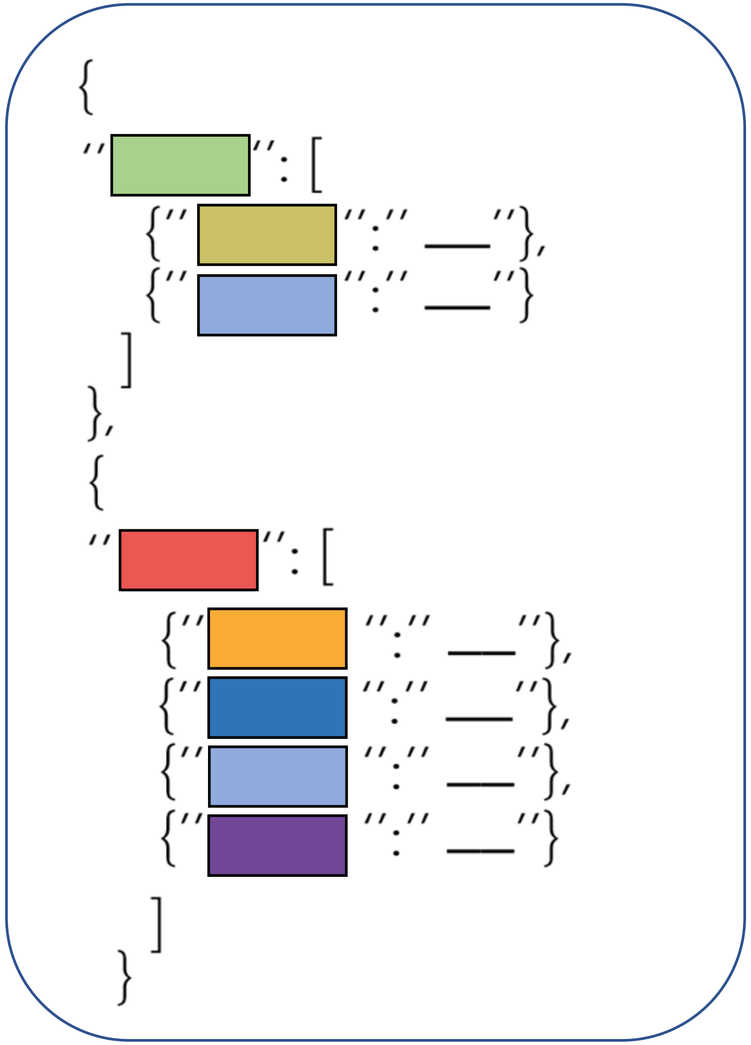
XMLParser

Documents can be very complex, and you can extract only required data set

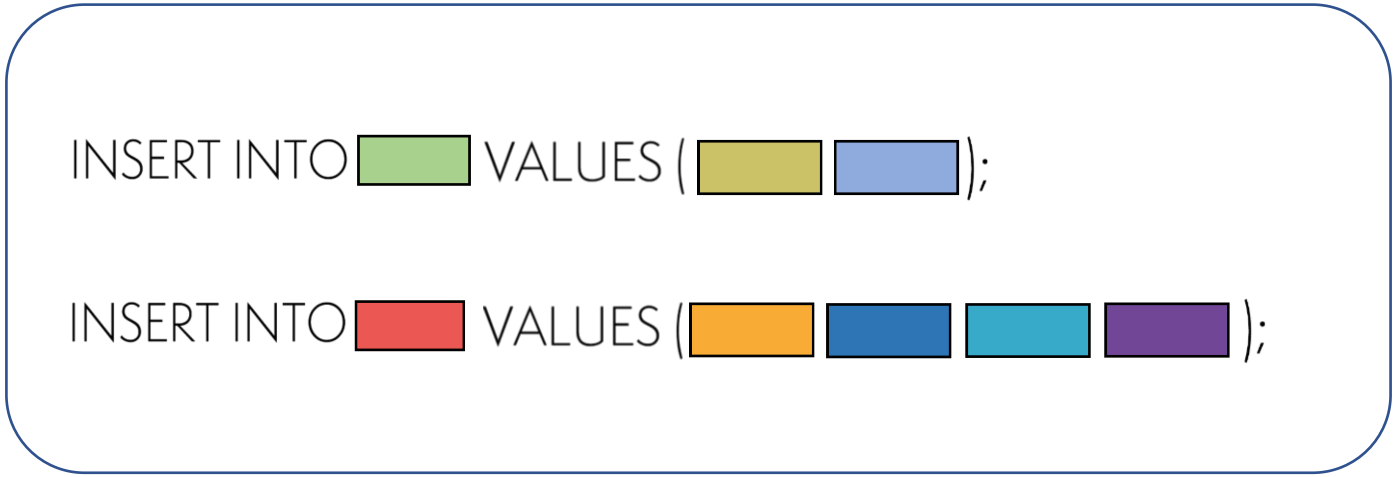


XMLParser

Every XML can be represented as JSON or SQL



```
{
"🟩": [
  {"🟨":" ___"},
  {"🟦":" ___"}
 ]
},
{
"🟥": [
  {"🟧":" ___"},
  {"🟦":" ___"},
  {"🟦":" ___"},
  {"🟪":" ___"}
  ]
 }
```

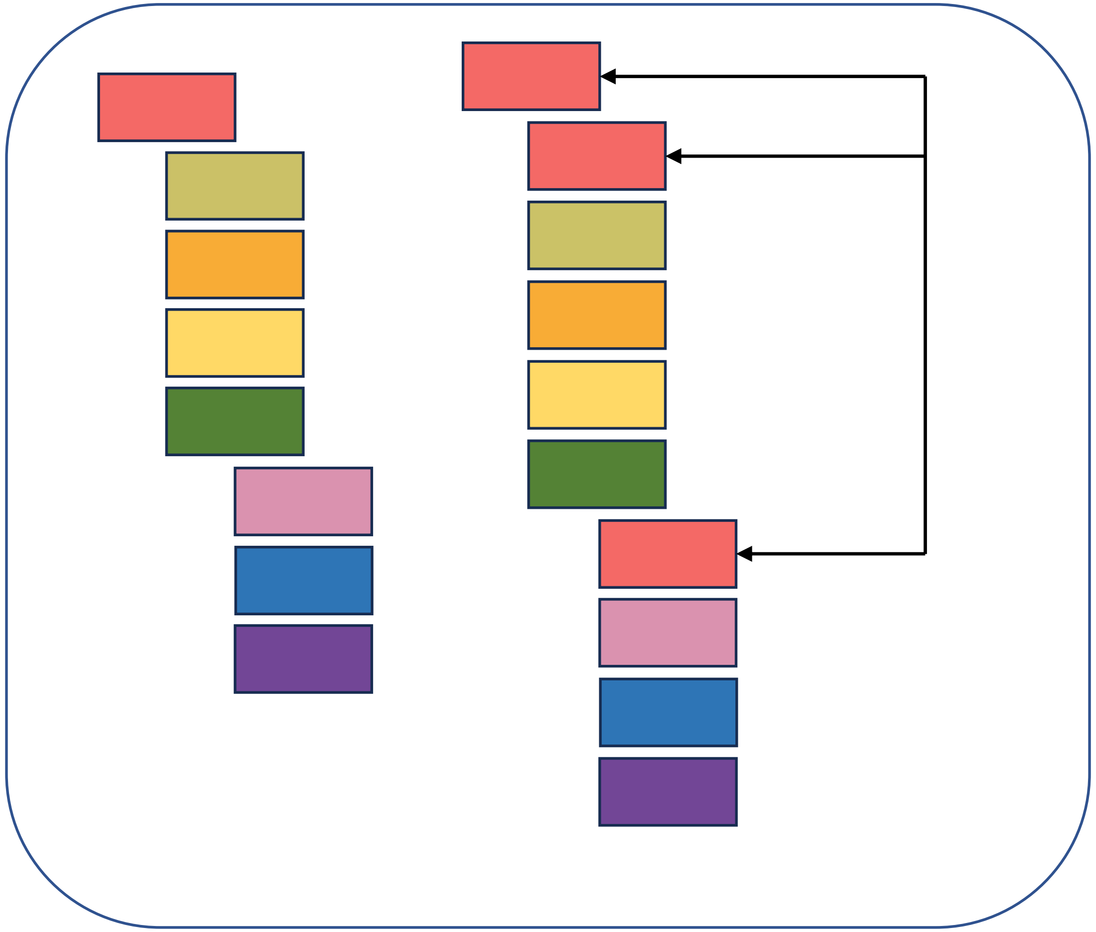


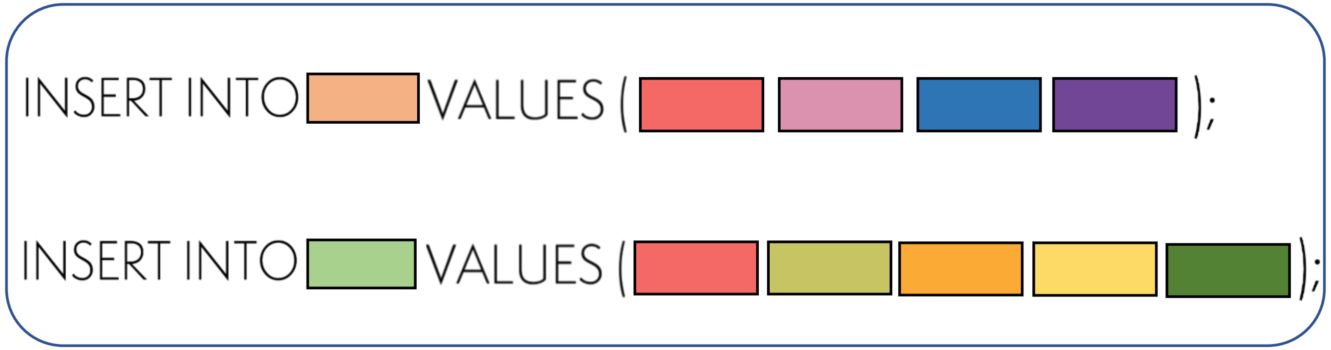
INSERT INTO 🟩 VALUES (🟨 🟦);

INSERT INTO 🟥 VALUES (🟧 🟦 🟦 🟪);

XMLParser

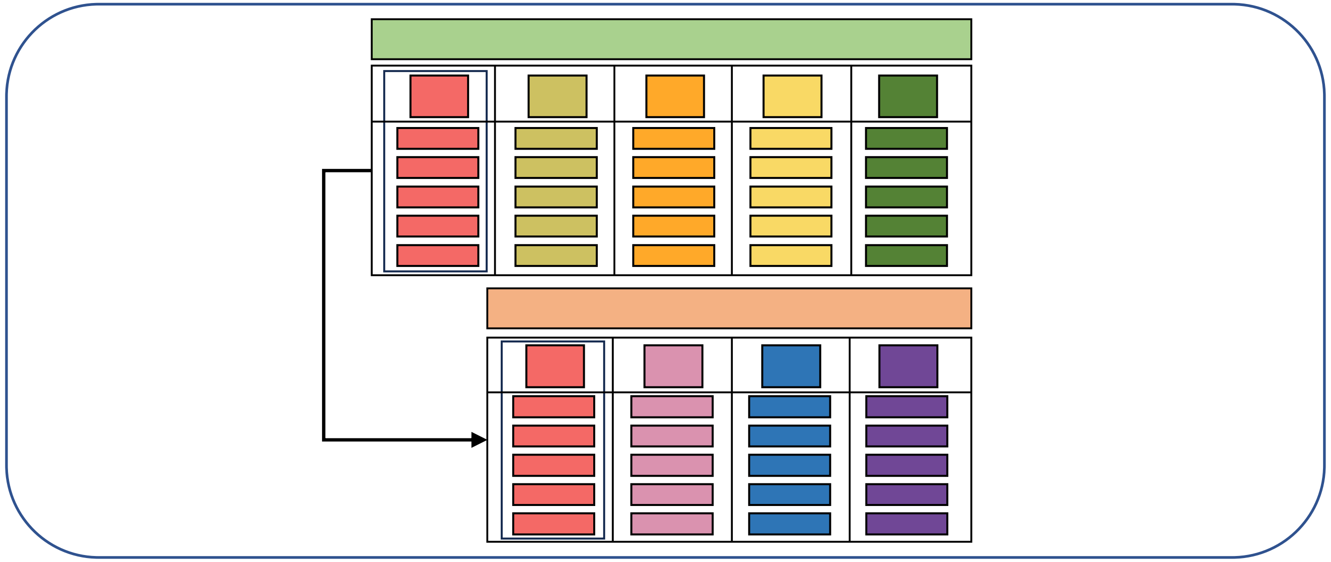
When generating SQL, you can preserve the parent-child relation





INSERT INTO VALUES ( );

INSERT INTO VALUES ( );



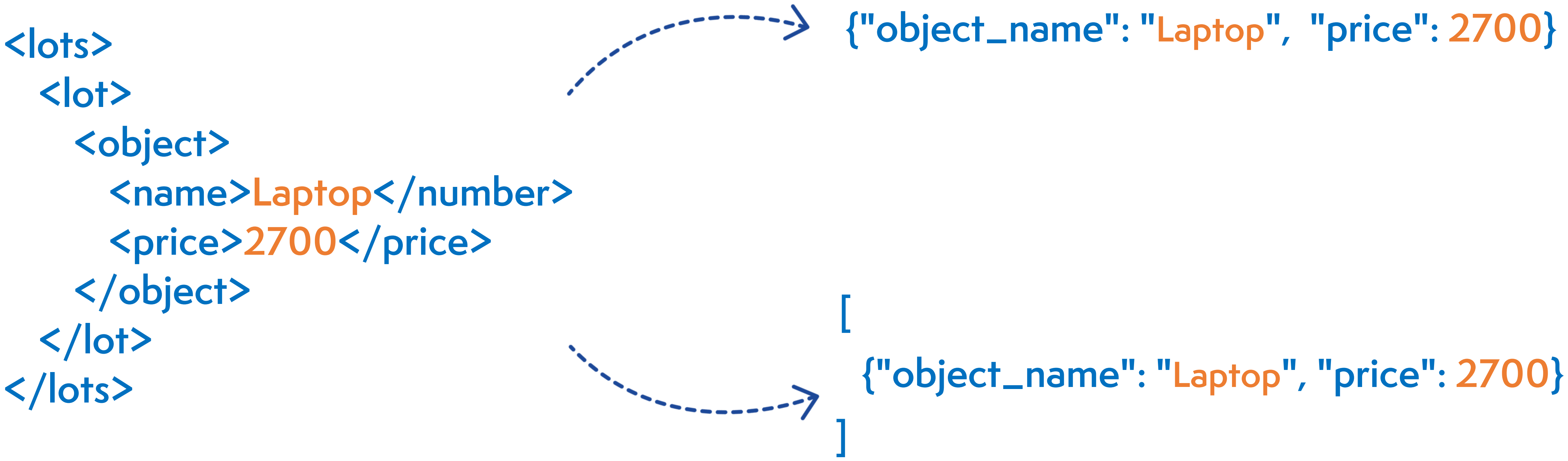
XMLParser

You can flexibly customize the SQL generation

- You can set table constrains and generate ON CONFLICT DO UPDATE queries to prevent rows duplications;
- You can insert multiple VALUES in single query;
- You can use some documents for updating already exists records in DB;
- You can generate plain text SQL for debugging or for other purposes.

XMLParser

You can specify when to extract data as an object, and when as a list of objects



```
<lots>
  <lot>
    <object>
      <name>Laptop</number>
      <price>2700</price>
    </object>
  </lot>
</lots>
```

```
{"object_name": "Laptop",  "price": 2700}
```

```
[
  {"object_name": "Laptop", "price": 2700}
]
```

XMLParser

Extract values if they are placed as tag attributes in same way as there were separated tags:

```
<lots>
  <lot>
    <object>
      <name>Laptop 1</number>
      <price>1650</price>
    </object>
    <object>
      <name>Laptop 2</number>
      <price>2100</price>
    </object>
  </lot>
  <lot>
    <object>
      <name>Laptop 3</number>
      <price>2700</price>
    </object>
  </lot>
</lots>
```

=

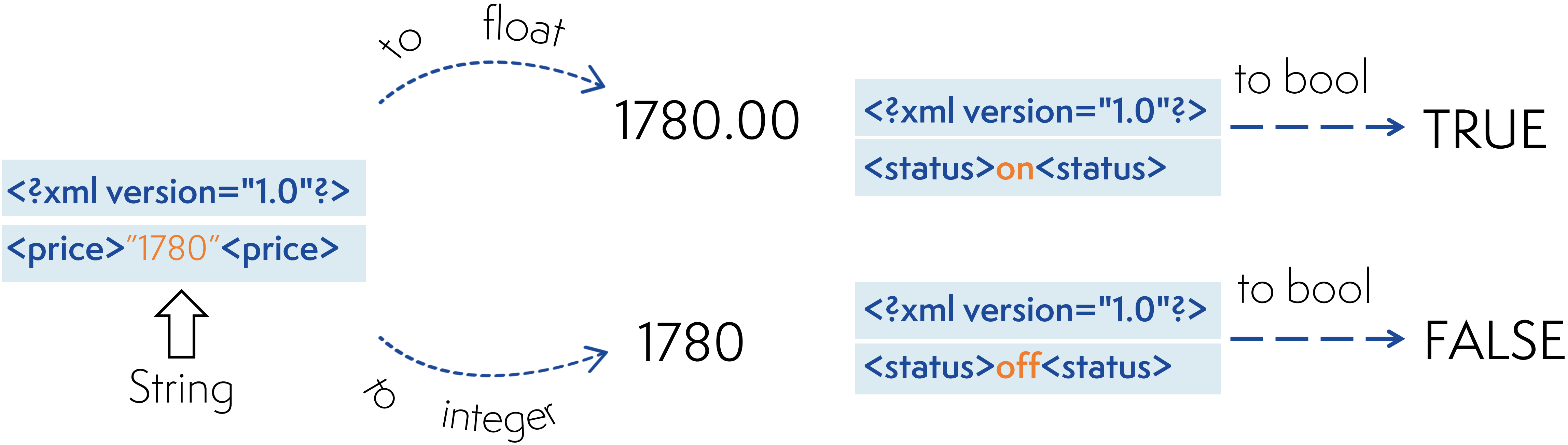
```
<lots>
  <lot>
    <object name=”Laptop 1” price=1650/>
  </lot>
  <lot>
    <object name=” Laptop 2” price=2100/>
  </lot>
  <lot>
    <object name=” Laptop 3” price=2700/>
  </lot>
</lots>
```

XMLParser

XSD schemes that never work?
Forget about them, you don't need them anymore!

XMLParser

You can do very tricky type conversion



```
<?xml version="1.0"?>
<price>"1780"<price>
```

String

to float

1780.00

to integer

1780

```
<?xml version="1.0"?>
<status>on<status>
```

to bool

TRUE

```
<?xml version="1.0"?>
<status>off<status>
```

to bool

FALSE

XMLParser

In more complex cases you may use build in Natural Language Processing.
It's very tiny but with it you can for example
evaluate some text as logical expression and store in database only
true/false flag instead of full text:

```
 <id>1</id>
<order>
  <status>the order was canceled</status>
</order>
```

OR

```
 <id>1</id>
<order>
  <status>canceled by user</status>
</order>
```

id	order_canceled
1	TRUE

XMLParser

In more complex cases you may use build in Natural Language Processing.
It's very tiny but with it you can for example
parse text (if it have some regular patterns) and place some it's parts as
separate fields:

```
<order>
  <address>country: Netherlands city: Amsterdam, phone: +31620874518</address>
</order>
```

country	city	phone
Netherlands	Amsterdam	+31620874518

XMLParser

In more complex cases you may use build in Natural Language Processing.
It's very tiny but with it you can for example for
do some data cleaning/filtering/masking

Symbol of dollar removed
to allow place data to numeric field:

```
<order>
   <price>$1780</price>
</order>
```

currency	price
USD	1780

Card number was masked:

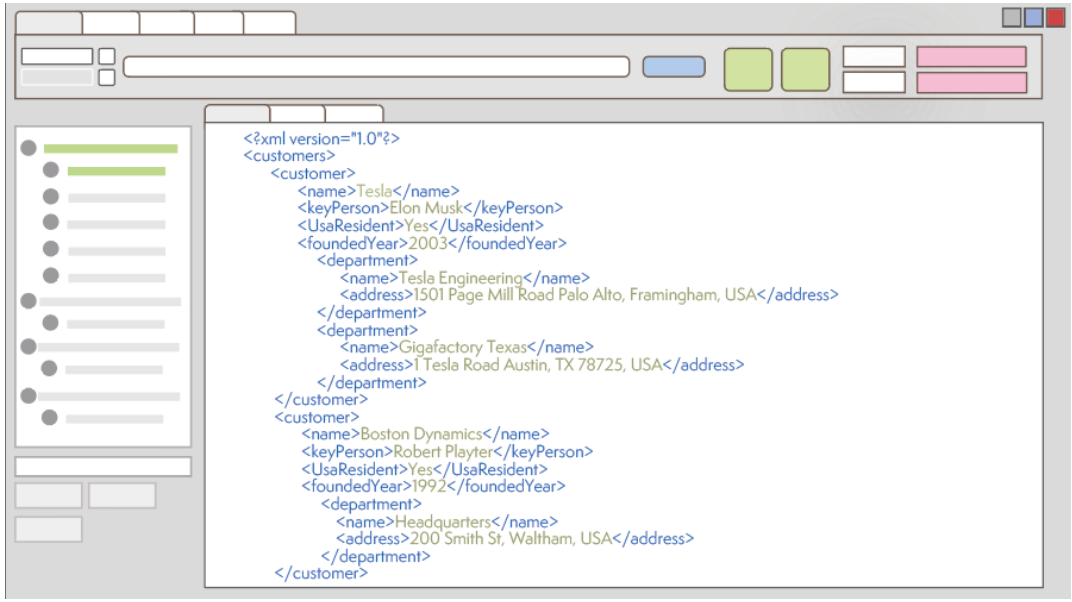
```
<order>
   <bankCardNumber>4716957520893445</bankCardNumber>
</order>
```

id	cardNumber
1	47**********3445

XMLParser

Easy way to parse XML and load data into SQL/NoSQL database.

+ Flexible and lightweight;
+ Easy to scale and suitable for big data;
+ Not a cloud solution. Your data remains private and local;
+ Doesn't require knowledge of programming;
+ Built-in tiny Natural Language Processing engine for complex rules;
+ Supports multi-model data representation.



```
<?xml version="1.0"?>
<customers>
    <customer>
        <name>Tesla</name>
        <keyPerson>Elon Musk</keyPerson>
        <UsaResident>Yes</UsaResident>
        <foundedYear>2003</foundedYear>
          <department>
            <name>Tesla Engineering</name>
            <address>1501 Page Mill Road Palo Alto, Framingham, USA</address>
          </department>
          <department>
            <name>Gigafactory Texas</name>
            <address>1 Tesla Road Austin, TX 78725, USA</address>
          </department>
    </customer>
    <customer>
        <name>Boston Dynamics</name>
        <keyPerson>Robert Playter</keyPerson>
        <UsaResident>Yes</UsaResident>
        <foundedYear>1992</foundedYear>
          <department>
            <name>Headquarters</name>
            <address>200 Smith St, Waltham, USA</address>
          </department>
    </customer>
```

bubnenkoff@gmail.com