

JSON data calculation and importing into database





CONTENTS

1

Import and parse

2

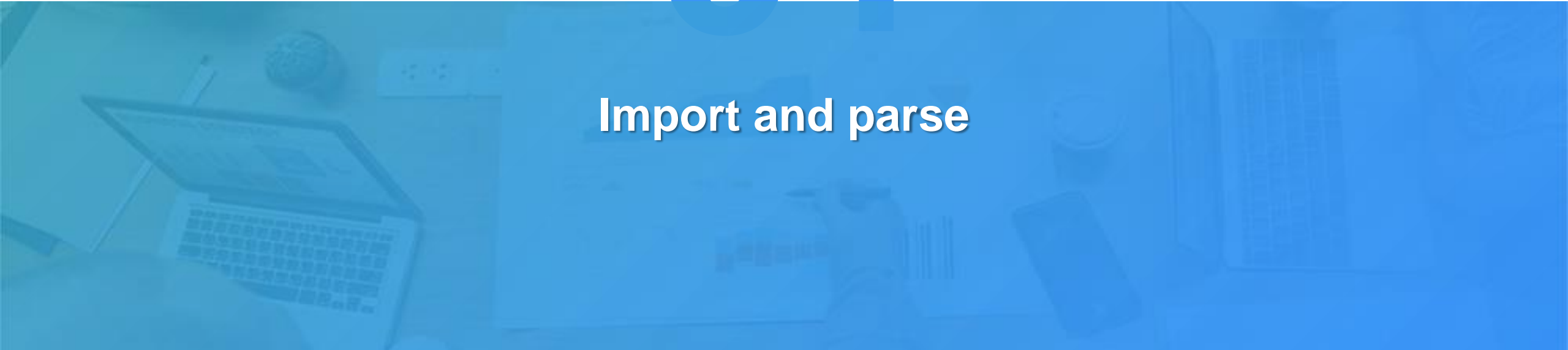
Import sequence table into
database

3

Summary

01

Import and parse



1. Single layer JSON data file

Let's start with a simple example to see how to read the JSON file of common key value mapping. The following is the JSON data of product information:

```
[{"PRODUCT_ID":1,"PRODUCT_NAME":"Apple Juice",  
"SUPPLIER_ID":2,"CATEGORY_ID":1, ...},  
{"PRODUCT_ID":2,"PRODUCT_NAME":"Milk",  
"SUPPLIER_ID":1,"CATEGORY_ID":1, ...},  
{"PRODUCT_ID":3,"PRODUCT_NAME":"Tomato sauce",  
"SUPPLIER_ID":1,"CATEGORY_ID":2, ...},  
{"PRODUCT_ID":4,"PRODUCT_NAME":"Salt",  
"SUPPLIER_ID":2,"CATEGORY_ID":2, ...},  
...]
```

1. Single layer JSON data file

It only needs a simple one sentence script for SPL to import JSON data file:

```
=json(file("product.json").read())
```

The results is as follows:

PRODUCT_ID	PRODUCT_NAME	SUPPLIER_ID	CATEGORY_ID	...
1	Apple Juice	2	1	...
2	milk	1	1	...
3	Tomato sauce	1	2	...
4	salt	2	3	...
...

2. Multi layer JSON data file with the same structure of detail data

Here is the JSON data for the order information. There are two layers: the first is country and area, and the second is detailed data. Now we want to import orders from North and South China in 2013.

```
[{"COUNTRY":"China","AREA":"Northeast China","ORDERS":[
{"ORDER_ID":10252,"CUSTOMER_ID":"SUPRD","EMPLOYEE_ID":4, ...},
{"ORDER_ID":10318,"CUSTOMER_ID":"ISLAT","EMPLOYEE_ID":8, ...},
...]},
{"COUNTRY":"China","AREA":"East China","ORDERS":[
{"ORDER_ID":10249,"CUSTOMER_ID":"TOMSP","EMPLOYEE_ID":6, ...},
{"ORDER_ID":10251,"CUSTOMER_ID":"VICTE","EMPLOYEE_ID":3, ...},
...]},
...]
```

2. Multi layer JSON data file with the same structure of detail data

Define parameters: country, area, and year. In the future, when importing different countries, areas, and years, you no longer need to modify SPL, just modify the corresponding parameter values. It should be noted that the value of area is a sequence, so that the data of multiple areas can be read at the same time. As in the following figure:

Name	Value
Country	China
Area	[North China, South China]
Year	2013

2. Multi layer JSON data file with the same structure of detail data

Let's take a look at the SPL script:

	A	B
1	<code>=json(file("orders.json").read())</code>	<code>=A1.select(COUNTRY==Country && Area.contain(AREA))</code>
2	<code>=B1.news(ORDERS;COUNTRY, AREA,\${B1.ORDERS.fname().concat@ c()})</code>	<code>=A2.select(year(ORDER_DATE)==Year)</code>

2. Multi layer JSON data file with the same structure of detail data

```
=json(file("orders.json").read())
```

First, import the JSON file. The data is multi-layer:

COUNTRY	AREA	ORDERS
China	Northeast China	[[10252,SUPRD,4,...],[10315,ISLAT,4,...],...]
China	East China	[[10249,TOMSP,6,...],[10251,VICTE,3,...],...]
China	Central China	[[10254,CHOPS,5,...],[10265,BLONP,2,...],...]
China	North China	[[10248,VINET,5,...],[10250,HANAR,4,...],...]
China	South China	[[10287,RICAR,8,...],[10296,LILAS,6,...],...]



Double click to view details

ORDER_ID	CUSTOMER_ID	EMPLOYEE_ID	...
10287	RICAR	8	...
10296	LILAS	6	...
...

2. Multi layer JSON data file with the same structure of detail data

The year and area fields are on the first level, and we can directly filter out the data of North and South China.

```
=A1.select(COUNTRY==Country && Area.contain(AREA))
```

The result is as follows:

COUNTRY	AREA	ORDERS
China	North China	[[10248,VINET,5,...],[10250,HANAR,4,...],...]
China	South China	[[10287,RICAR,8,...],[10296,LILAS,6,...],...]

2. Multi layer JSON data file with the same structure of detail data

```
=B1.news(ORDERS;COUNTRY, AREA,${B1.ORDERS.fname().concat@c()})
```

The filtered result is used to generate a sequence table, which consists of the fields of country, area and order details. The results are as follows:

COUNTRY	AREA	ORDER_ID	CUSTOMER_ID	EMPLOYEE_ID	ORDER_DATE
China	North China	10248	VINET	5	2012-07-04
China	North China	10250	HANAR	4	2012-07-08
China	North China	10253	HANAR	3	2012-07-10
China	North China	10255	RICSU	9	2012-07-12

Here, the parameters of the news function use macros. Macro uses `${}` to enclose expressions. SPL will first evaluate macro expressions, and then replace `${}` with the evaluated results as string values. The actual execution of A2 is:

```
=B1.news(ORDERS;COUNTRY, AREA, ORDER_ID, CUSTOMER_ID, EMPLOYEE_ID, ORDER_DATE, ...)
```

2. Multi layer JSON data file with the same structure of detail data

Finally, the records with order date as year 2013 are selected from the sequence table.

```
=A2.select(year(ORDER_DATE)==Year)
```

The final result is as follows:

COUNTRY	AREA	ORDER_ID	CUSTOMER_ID	EMPLOYEE_ID	ORDER_DATE
China	North China	10402	ERNSH	8	2013-01-02
China	North China	10403	ERNSH	4	2013-01-03
China	North China	10404	MAGAA	2	2013-01-03
China	North China	10407	OTTIK	2	2013-01-07

3. Multi layer JSON data files with different structure of detailed data

Because of the complexity of data sources, the detailed data of JSON data files may be of different structures. In the following sales data: the first level takes year and month as the dimension, the second level takes country as the dimension, and the third level is detailed data. But in the detailed data, due to different sales channels, the data structure is not completely consistent. Now we're going to read sales data for 2017 and 2018 in the US and Canada.

```
[{"YEAR":2016,"MONTH":1,"SALES":  
 [{"COUNTRY":"Germany","SALES":  
 [{"ORDERNUMBER":10101,"QUANTITYORDERED":25,"PRICEEACH":100,"ORDE  
RLINENUMBER":4,"SALES":3782,"ORDERDATE":"1/9/2016 0:00", ...}, ...], ...],  
 {"YEAR":2016,"MONTH":2,"SALES":  
 [{"COUNTRY":"Denmark","SALES":  
 [{"ORDERNUMBER":10105,"QUANTITYORDERED":50,"PRICEEACH":100,"ORDE  
RLINENUMBER":2,"SALES":7208,"ORDERDATE":"2/11/2016 0:00", ...}, ...], ...],  
 ...]
```

3. Multi layer JSON data files with different structure of detailed data

We first need to determine the structure of the detailed data. In this example, we want to list all the fields. If the detailed data does not contain this field, it is set to blank. For example, the addressline2 field is missing from the following data:

YEAR	COUNTRY	ORDERNUMBER	ADDRESSLINE1	ADDRESSLINE2
2017	USA	10353	2440 Pompton St.	
2017	USA	10352	16780 Pompton St.	

For convenience, let's define two parameters first: year and country

Name	Value
Year	[2017, 2018]
Country	[USA, Canada]

3. Multi layer JSON data files with different structure of detailed data

Now let's look at SPL:

	A	B
1	<code>=json(file("sales.json").read())</code>	<code>=A1.select(Year.contain(YEAR))</code>
2	<code>=B1.news(SALES;YEAR,MONTH,COUNTRY,SALES)</code>	<code>=A2.select(Country.contain(COUNTRY))</code>
3	<code>for(B2)</code>	<code>=A3.SALES.fname()&B3</code>
4	<code>=B2.news(SALES; YEAR, COUNTRY,\${B3.concat@c()})</code>	

3. Multi layer JSON data files with different structure of detailed data

```
=json(file("sales.json").read())
```

```
=A1.select(Year.contain(YEAR))
```

First, import JSON file of multiple layers. Since the year field is on the first level, we can directly filter out the data of Year 2017 and 2018:

YEAR	MONTH	SALES
2017	1	[[France,[10211,41,100, ...],[10211,41,100, ...], ...], ...]
2017	2	[[Australia,[10223,37,100, ...],[10223,47,100, ...], ...], ...]
2017	3	[[Australia,[10227,25,100, ...],[10227,31,48.52, ...], ...], ...]
2017	4	[[Canada,[10235,24,76.03, ...],[10235,23,96.29, ...], ...], ...]
2017	5	[[Finland,[10247,44,100, ...],[10247,25,100, ...], ...], ...]

3. Multi layer JSON data files with different structure of detailed data

```
=B1.news(SALES;YEAR,MONTH,COUNTRY,SALES)
```

Use the news function to combine the year / month fields with the country and monthly sales details.

YEAR	MONTH	COUNTRY	SALES
2017	1	France	[[10211,41,100, ...],[10211,41,100, ...], ...]
2017	1	Japan	[[10210,23,100, ...],[10210,34,100, ...], ...]
2017	1	Spain	[[10212,39,100, ...],[10212,33,100, ...], ...]
2017	1	UK	[[10213,38,94.79, ...],[10213,25,83.39, ...], ...]
2017	1	USA	[[10215,35,100, ...],[10209,39,100, ...], ...]

3. Multi layer JSON data files with different structure of detailed data

B2: Filter out data of US and Canada through A2.select(Country.contain(COUNTRY)).

A3~A4: Because the detail data may have different structures, we use full fields names as parameters to create a sequence table. The value of the field will be set according to the name, and null will be set by default if the field does not exist. (As the "ADDRESSLINE1" and "ADDRESSLINE2" fields in the following figure) :

YEAR	COUNTRY	ORDERNUMBER	ADDRESSLINE1	ADDRESSLINE2
2017	USA	10353	2440 Pompton St.	
2017	USA	10352	16780 Pompton St.	
2017	USA	10352	16780 Pompton St.	
2018	USA	10369		
2018	USA	10362		
2018	USA	10371		

3. Multi layer JSON data files with different structure of detailed data

The final result:

YEAR	COUNTRY	ORDERNUMBER	QUANTITYORDERED	PRICEEACH	ORDERLINENUMBER
2017	USA	10215	35	100	3
2017	USA	10209	39	100	8
2017	USA	10215	46	100	2
2017	USA	10215	27	89.38	10
2017	USA	10215	33	43.13	9
2017	USA	10215	49	100	4

At this point, a JSON file with different detailed data structure of multi-layer structure is expanded into a two-dimensional table.

02

Import sequence table into database




1. Import single table into database

Take the JSON file of product order information in 1.1 as an example to update the parsed sequence table to the product table of the database.

JSON file:

```
[{"PRODUCT_ID":1,"PRODUCT_NAME":  
"Apple Juice",  
"SUPPLIER_ID":2,"CATEGORY_ID":1, ...},  
{"PRODUCT_ID":2,"PRODUCT_NAME":  
"Milk",  
"SUPPLIER_ID":1,"CATEGORY_ID":1, ...},  
{"PRODUCT_ID":3,"PRODUCT_NAME":  
"Tomato sauce",  
"SUPPLIER_ID":1,"CATEGORY_ID":2, ...},  
...]
```

Database table:



Product
PRODUCT_ID
PRODUCT_NAME
SUPPLIER_ID
CATEGORY_ID
...

1. Import single table into database

It's very simple for SPL to import sequence table into database, just using `db.update()` function. The SPL script is as follows:

	A
1	<code>=json(file("product.json").read())</code>
2	<code>=connect("db")</code>
3	<code>=A2.update(A1, Product)</code>

A1: Import JSON file as sequence table.

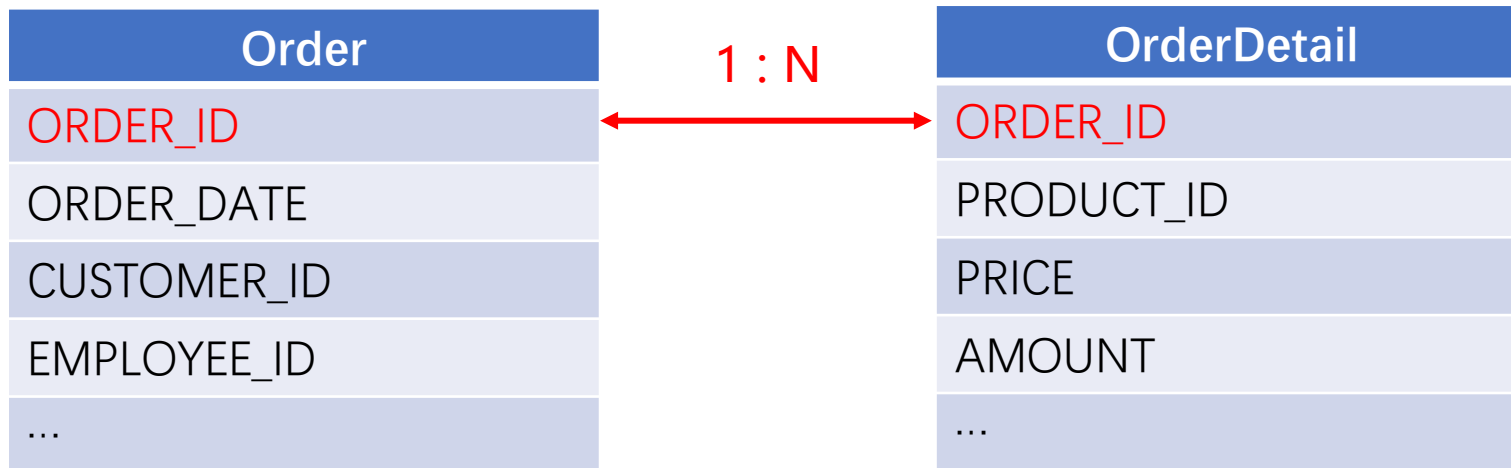
A2: Connect data source.

A3: Use `db.update()` function, update the sequence table imported in A1 to the product table in the database. Note that the primary key parameter of the update function is omitted here. In this case, it will be updated according to the primary key of the database table product; if the product table has no primary key, it will be updated according to the primary key of A1; if there is no primary key, it will be updated according to the first field.

2. Import multi tables into database

Take the JSON file of order information as an example. JSON data is divided into two layers: the first layer is order, and the second layer is order details. To update the order and order details of 2018 and later to the order table and order details table of the database respectively.

```
[{"ORDER_ID":10248,"ORDER_DATE":"2012-07-04",...,"ORDER_DETAILS":[{"PRODUCT_ID":17,"PRICE":14,"AMOUNT":12, ...}, {"PRODUCT_ID":42,"PRICE":9,"AMOUNT":9, ...}, ...]}, ...]
```



2. Import multi tables into database

Let's take a look at SPL, as follows:

	A	B
1	<code>=json(file("orders.json").read())</code>	<code>=A1.select(year(ORDER_DATE)>=2018)</code>
2	<code>=connect("demo")</code>	
3	<code>=B1.fname().delete(B1.fname().len())</code>	<code>=A2.update(B1,Order,\${A3.concat@c()})</code>
4	<code>=B1.conj(ORDER_DETAILS.derive(B1.ORDER_ID:ORDER_ID))</code>	<code>=A2.update(A4,OrderDetail)</code>

2. Import multi tables into database

```
=json(file("orders.json").read())
```

First, import the JSON file. The data is two-tier:

ORDER_ID	ORDER_DATE	CUMSTOMER_ID	ORDER_DETAILS
10248	2012-07-04	VINET	[[17,14,12,...],[42,9,10,...],...]
10249	2012-07-05	TOMSP	[[14,18,9,...],[51,42,4,...],...]
10250	2012-07-08	HANAR	[[41,7,10,...],[51,42,3,...],...]
10251	2012-07-08	VICTE	[[22,16,6,...],[57,15,15,...],...]
10252	2012-07-09	SUPRD	[[20,64,40,...],[33,2,25,...],...]



Double click to view details

PRODUCT_ID	PRICE	AMOUNT	...
20	64	40	...
33	2	25	...
...

2. Import multi tables into database

The order date field is on the first level, so the data of 2018 and later can be directly filtered out.

```
=A1.select(year(ORDER_DATE)>=2018)
```

The result is as follows:

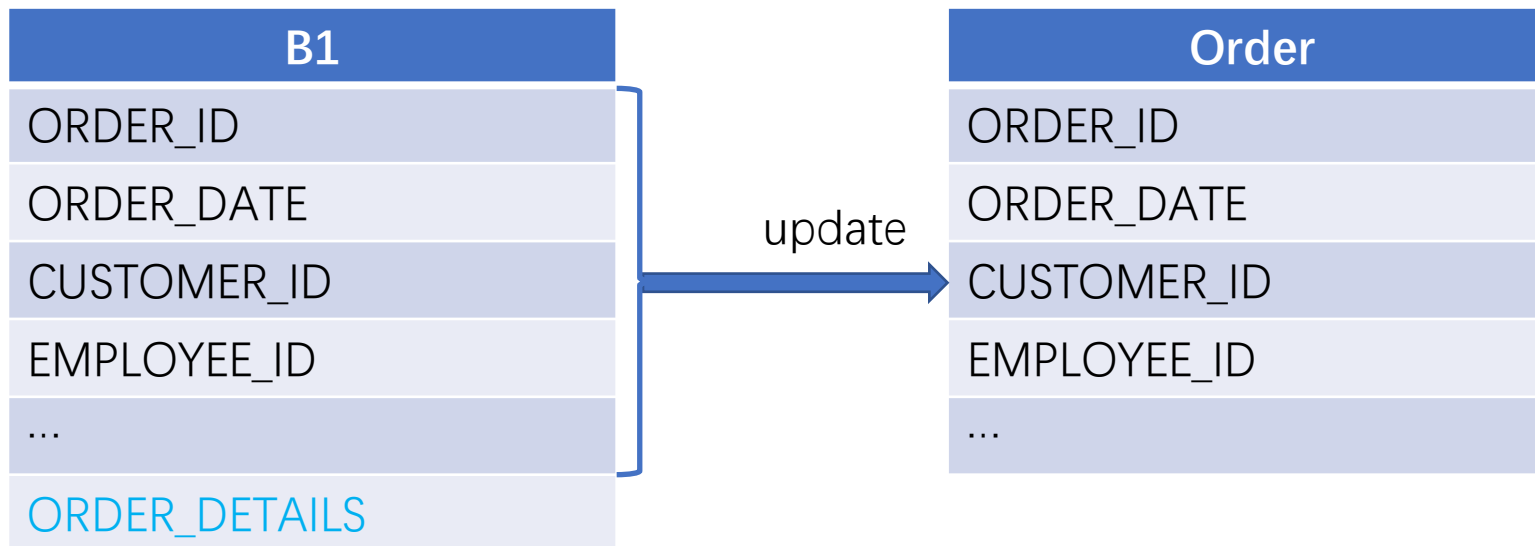
ORDER_ID	ORDER_DATE	CUMSTOMER_ID	ORDER_DETAILS
10808	2018-01-01	OLDWO	[[56,38,20,...],[76,18,50,...]]
10809	2018-01-01	WELLI	[[52,7,20,...]]
10810	2018-01-01	LAUGB	[[13,6,7,...],[25,14,5,...],...]

2. Import multi tables into database

```
=B1.fname().delete(B1.fname().len())
```

```
=A2.update(B1,Order,${A3.concat@c()})
```

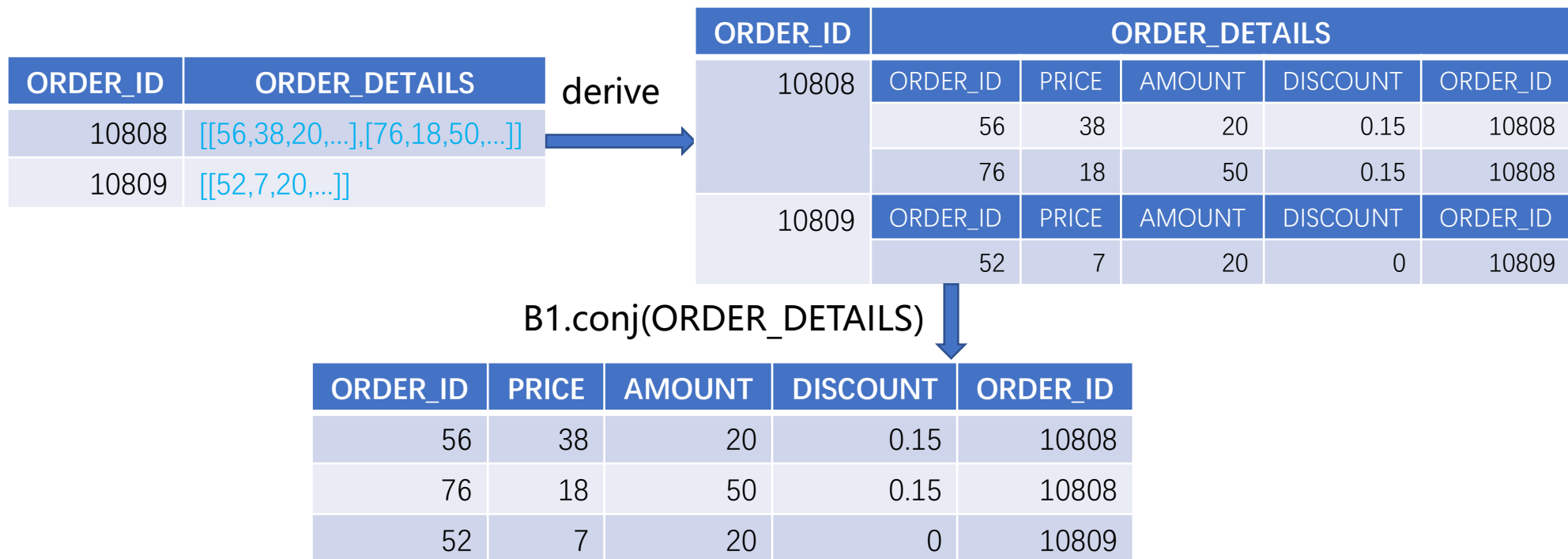
After connecting to the database, first update the main table order table. However, B1 has one more order_details field than the database table. When using the update function to update, you need to specify the update field. The field parameter uses a macro. Use B1.fname() to get all the field names, and then delete the last member.



2. Import multi tables into database

```
=B1.conj(ORDER_DETAILS.derive(B1.ORDER_ID:ORDER_ID))
```

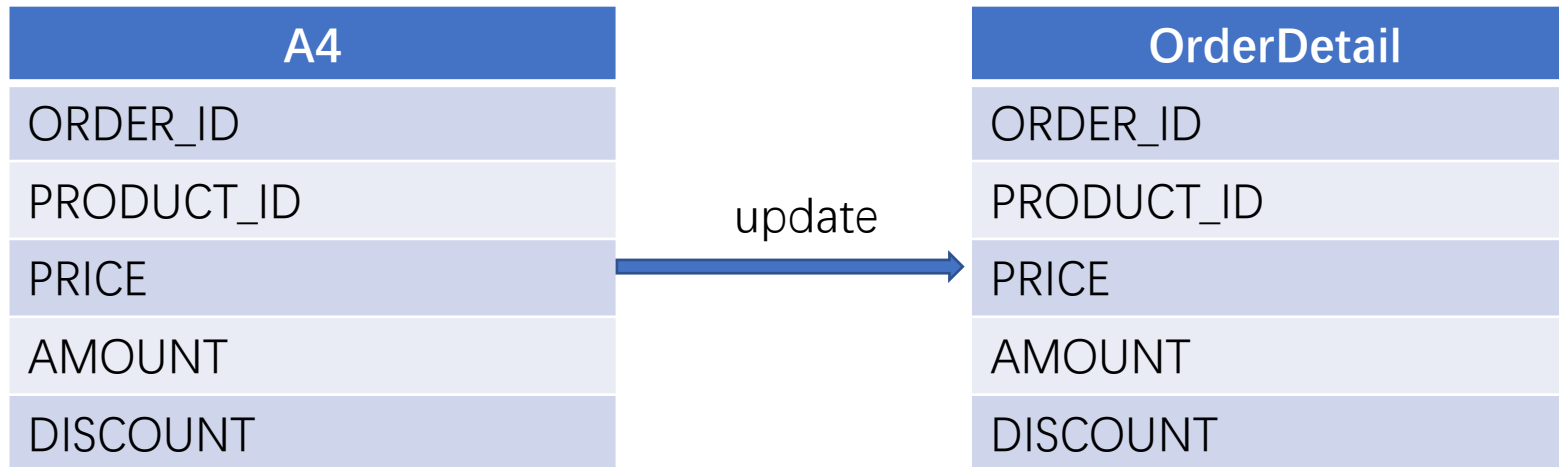
First, use the `derive()` function to add the order ID field to `order_details`. Then, using the function of `conj()`, the `order_details` of each order is expanded and put together, which is consistent with the data structure of the order details in the database. The results are as follows:



2. Import multi tables into database

```
=A2.update(A4,OrderDetail)
```

Finally, update the sub table order details. Since the data structure of A4 is consistent with the database table, it is no longer necessary to specify update fields.



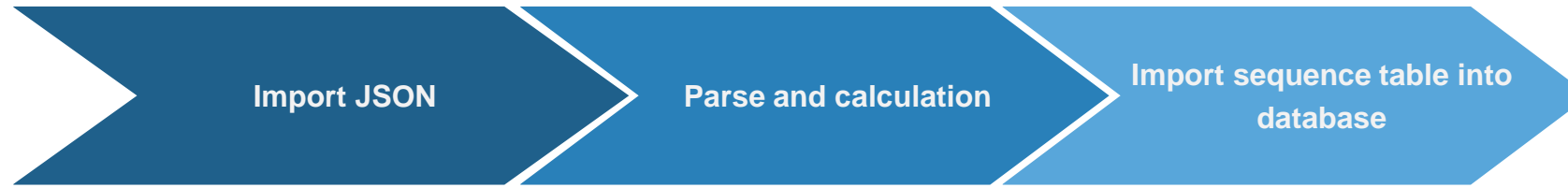
03

Summary



Summary

Process of using JSON data



First read in the JSON file, and then use the JSON () function to parse it into a sequence table.

According to the actual needs, analyze and calculate the data. SPL sequence table provides a wealth of functions, which can be used for various operations.

Just use the db.update() function to import the sequence table into database. When JSON files correspond to multiple tables, pay attention to the order of update, from the main table to the sub table.

As we can see from the previous chapters, the focus of JSON data usage is the parsing and calculation part. When dealing with multi-layer and complex data with different structure, SPL can simply use "table. field" to reference members, and there are rich functions to support calculation.



THANKS