

Transposition



CONTENTS

1

Row to column

2

Column to row

3

Bidirectional transposition

4

Dynamic row and column
transposition

5

Inter-column calculation at
transposition

6

Inter-table join column to row
conversion

7

Divided by columns

01

Row to column

A blue-tinted background image of a desk. On the left, a laptop is open, displaying a grid-like pattern. In the center, a hand is holding a pen over a document. To the right, there is a smartphone and a small container. The overall scene is a workspace or office environment.

Pivot of database

The following is the student scores table. The highest scores of each subject in each class are counted and displayed by column.

Class	StudentID	Subject	Score
Class one	1	Math	89
Class one	1	Chinese	93
Class two	2	Math	92
Class two	2	Chinese	97



Pivot function of database supports row to column conversion

Class	MathMax	ChineseMax
Class one	89	93
Class two	92	97

Pivot of database

The SQL statement is as follows:

```
select * from ( select Class, Subject, Score from StudentScore )
pivot (
    max(Score) for Subject in (
        'Math' as MathMax, 'Chinese' as ChineseMax
    )
)
```

Here is an example of Oracle. Not all databases have pivot functions. Pivot is only supported in new versions of mainstream databases.

Pivot of SPL

First, get the highest scores of each subject from the database.

Class	Subject	MaxScore
Class one	Math	89
Class one	Chinese	93
Class two	Math	92
Class two	Chinese	97



Then use pivot function of SPL to make row to column conversion.

Class	MathMax	ChineseMax
Class one	89	93
Class two	92	97

Pivot of SPL

The SPL script is as follows:

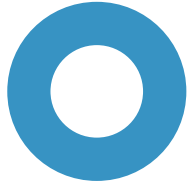
	A
1	=connect("oracle")
2	=A1.query("select Class, Subject, max(Score) MaxScore from StudentScore group by Class, Subject")
3	=A2.pivot(Class; Subject, MaxScore; "Math":"MathMax", "Chinese":"ChineseMax")

A1: Connect database.

A2: Fetch data from database, and the highest scores of each subject are calculated directly here.

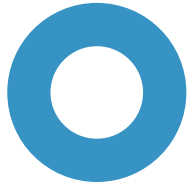
A3: The pivot function is used to realize row to column conversion.

Row to column transposition summary



Not all databases support pivot.

For example, MySQL does not have pivot function. It needs to use sub query to group and aggregate, then left join to realize row to column conversion.



SPL's pivot is more adaptable.

It is difficult for SQL statement to handle the situation that inter column calculation is needed after row to column conversion. SQL cannot solve the situation of multiple data sources.

02

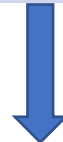
Column to row



Unpivot of SQL

The following is the student scores table. Query the highest score of the students.

StudentID	Math	Chinese
1	89	93
2	92	97
3	91	88



The unpivot function of database supports column to row conversion.

StudentID	BestScore
1	93
2	97
3	91

Unpivot of SQL

Take Oracle as an example. The SQL statement is as follows:

With

```
T1 as (select * from StudentScore unpivot (Score for Subject in  
(Math, Chinese)))
```

```
select StudentID, max(T1.Score) BestScore from T1 group by  
StudentID
```

Column to row conversion of SPL

Column to row conversion of SPL looks clearer. The first step is to turn the column into a row, and the second step is to get the highest score of each student by grouping and aggregation.

StudentID	Math	Chinese
1	89	93
2	92	97
3	91	88

StudentID	BestScore
1	93
2	97
3	91

StudentID	Subject	Score
1	Math	89
1	Chinese	93
2	Math	92
2	Chinese	97
3	Math	91
3	Chinese	88

Column to row conversion of SPL

The SPL script is as follows:

	A
1	=connect("oracle")
2	=A1.query("select * from StudentScore")
3	=A2.pivot@r(StudentID; Subject, Score; Math:"Math", Chinese:"Chinese")
4	=A3.groups(StudentID; max(Score):BestScore)

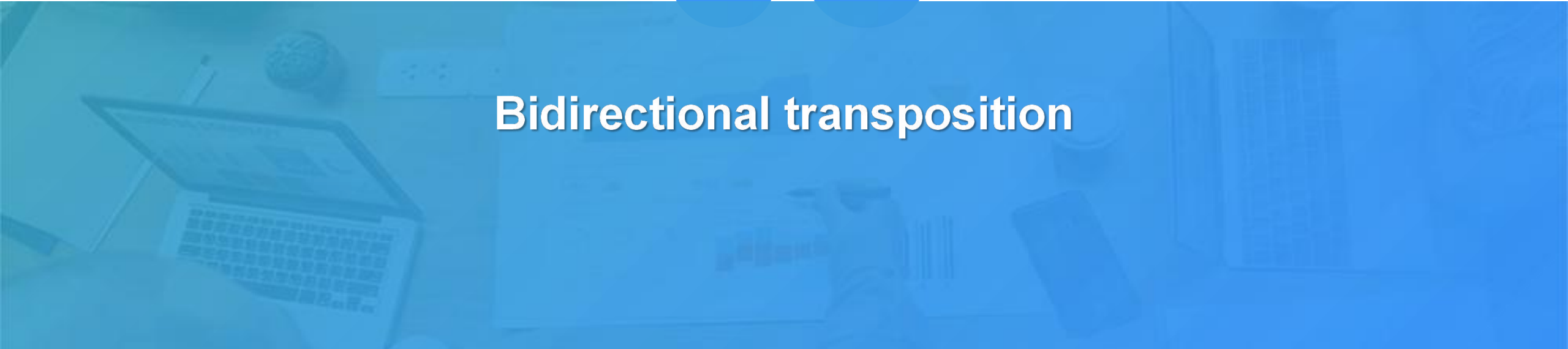
A1: Connect database. A2: Read student score table.

A3: Use the @r option of the pivot function to implement column to row transposition.

A4: For the data after transposition, the maximum value is taken after grouping by student ID.

03

Bidirectional transposition



Bidirectional transposition

The sales table classified by channel and recorded by date is as follows:

Day	Online	Store
20190101	2400	1863
20190102	1814	670
20190103	3730	1444


Need to convert to the following:

Category	20190101	20190102	20190103
Online	2400	1814	3730
Store	1863	670	1444

Bidirectional transposition


First, the columns are converted to rows, converting online and store to field values for category column.

Day	Online	Store
20190101	2400	1863
20190102	1814	670
20190103	3730	1444



Day	Category	Sales
20190101	Online	2400
20190101	Store	1863
20190102	Online	1814
20190102	Store	670
20190103	Online	3730
20190103	Store	1444

Then row to column conversion, the unique value of the day field value is converted to the column names.



Category	20190101	20190102	20190103
Online	2400	1814	3730
Store	1863	670	1444

Bidirectional transposition

The SPL script is as follows:

	A
1	=connect("db")
2	=A1.query("select * from Sales")
3	=A2.pivot@r(Day; Category, Sales)
4	=A3.pivot(Category; Day, Sales)

A3: Use pivot@r to make column to row conversion, converting online and store to field values for category column.

A4: Use pivot to make row to column conversion, convert the unique value of the day field value to the column names.

Pivot function summary



Pivot function is suitable for static transposition

When the number of rows and columns does not change, it is convenient to use pivot function directly to implement transposition. When two-way transposition is needed, the pivot and pivot @r functions of SPL can solve the problem.

In the face of the dynamic transpose of uncertain table structure after transpose, pivot function is not competent. We will provide solutions in later chapters.

04

Dynamic row and column transposition



1. Pivot function generates columns automatically

There is an employee table, which records the Department, location and income of employees, as shown in the following figure:

Name	Dept	Area	Salary
David	Sales	Beijing	8000
Daniel	R&D	Beijing	15000
Andrew	Sales	Shanghai	9000
Robert	Sales	Beijing	26000
Rudy	R&D	Shanghai	23000

Calculate the average salary of each department in different regions. Now I don't know which regions, I want to convert it into the following results:

Dept	Beijing	Shanghai	...
Sales	13000	11000	...
R&D	15000	14000	...

1. Pivot function generates columns automatically

This example is row to column conversion. The target column needs to be extracted from the data. Pivot function supports semi dynamic transpose. When the target column (source column) is not specified, it will automatically find all columns not in the group.

The SPL script is as follows:

	A
1	=connect("db")
2	=A1.query("select Dept,Area,avg(Salary) as AvgSalary from Employee group by Dept,Area")
3	=A2.pivot(Dept; Area, AvgSalary)

A1: Connect data source.

A2: Take the average salary grouped by department and area from the employee table.

A3: The pivot function is used for row to column conversion, where the target column is omitted.

2. Dynamic row and column transposition

There is a personal income table, as shown in the following figure:

Name	Source	Income
David	Salary	8000
David	Bonus	15000
Daniel	Salary	9000
Andrew	Shares	26000
Andrew	Sales	23000
Robert	Bonus	13000

Everyone's income source may be different, and we want to convert it into the following results:

Category	Source1	Income1	Source2	Income2
David	Salary	8000	Bonus	15000
Daniel	Salary	9000		
Andrew	Shares	26000	Sales	23000
Robert	Bonus	13000		

2. Dynamic row and column transposition

We are not sure about the number of columns or even the names of columns after row to column conversion. At this point, you can't use the pivot function, but you need to use the dynamic transpose method.

The SPL script is as follows:

	A	B
1	=connect("db")	=A1.query("select * from Income")
2	=B1.group(Name)	=A2.max(~.len())
3	=create(Name, \${B2.("Source"+string(~)+"", "Income"+string(~)).concat@c()})	
4	for A2	=A4. Name A4.conj([Source, Income])
5		>A3.record(B4)

A3: determine the number of columns according to the maximum number of group members after grouping, and dynamically generate column names to create a sequence table.

A4~B5: Loop members of income table after grouping, put together the name, income source and income amount of each group, and add them to the sequence table created by A3.

Dynamic row and column transposition summary



Dynamic transpose calculates target data structure first

The key of dynamic transposition is to calculate the target data structure first. After the table structure is determined, the data is put together into records according to the data structure and inserted into the target table. This idea is also applicable to some static transposes.

3. Complex static row column transposition

Let's take another example. There is a table for recording daily attendance information, as shown in the following figure:

Per_Code	in_out	Date	Time	Type
1110263	1	2013-10-11	09:17:14.0000000	In
1110263	6	2013-10-11	11:37:00.0000000	Break
1110263	5	2013-10-11	11:38:21.0000000	Return
1110263	0	2013-10-11	11:43:21.0000000	NULL
1110263	6	2013-10-11	13:21:30.0000000	Break
1110263	5	2013-10-11	14:25:58.0000000	Return
1110263	2	2013-10-11	18:28:55.0000000	Out

Every seven pieces of data are in a group. We want to convert them into the following results:

Per_Code	Date	In	Out	Break	Return
1110263	2013-10-11	9:17:14	18:28:55	11:37:00	11:38:21
1110263	2013-10-11	9:17:14	18:28:55	13:21:30	14:25:58

3. Complex static row column transposition

Although the structure of the transposed table can be determined, it is very complex to implement it with pivot. You can create the target data structure first, and then fill in the data.

The SPL script is as follows:

	A	B
1	=connect("db").query("select * from DailyTime order by Per_Code,Date,Time")	=A1.group(Per_Code,Date)
2	=create(Per_Code,Date,In,Out,Break,Return)	=B1.([1,7,2,3,1,7,5,6].(B1.~(~)))
3	=B2.conj([~.Per_Code,~.Date] ~.(Time).m([1,2,3,4]) ~.Per_Code,~.Date] ~.(Time).m([5,6,7,8]))	>A2.record(A3)

A1: Query data and group by person code, date and time. B1: Group by person code and date.

A2: Create an empty sequence table to store the final result.

B2: For each group, take out the first, seventh, second, third, first, seventh, fifth and sixth records in turn, which is an orderly all day record.

A3: Organize all the data of each record into a sequence. B3: Add records to the sequence table created by A2.

4. Complex dynamic row column transposition

There are two tables, the user table and the record table, indicating that the user has an active record on a certain day. Two tables are joined by user ID, as in the following figure:



To display whether the user has a record of activities every week in 2018, as in the following table:

Week	User1	User2	User3
1	Yes	No	Yes
2	Yes	Yes	No
3	Yes	No	Yes
4	No	Yes	Yes

4. Complex dynamic row column transposition

This example looks a little more complicated, but the idea is the same. Create the target data structure first, and then fill in the data.

The SPL script is as follows:

	A	B
1	<code>=connect("db").query("select t1.ID as ID, t1.Name as Name, t2.Date as Date from User t1, Record t2 where t1.ID=t2.ID")</code>	
2	<code>=A1.derive(interval@w("2018-01-01",Date)+1:Week)</code>	<code>=A2.max(Week)</code>
3	<code>=A2.group(ID)</code>	<code>=B2.new(~:Week,\${A3.("\No\":"+Name).concat@c()})</code>
4	<code>=A3.run(~.run(B3(Week).field(A3.#+1,"Yes")))</code>	

A1: Query user table and record table, and join by user ID. A2: Calculate the week number according to the date.

B2: Find the largest week number. A3: Group by user ID.

B3: Create an empty sequence table according to the maximum weekly number, and assign the default value "no".

A4: For each group of data, locate the corresponding record in the target table through the weekly sequence number, and replace the user value with "yes".

05

Inter-column calculation at transposition



Inter-column calculation at transposition

The calculation process is still to generate an empty result set first and then append data. The difference is that the data to be appended here needs a series of calculations. The SPL script is as follows:

	A	B
1	=connect("db").query("select * from Payment.txt where year(due_date)=2014")	
2	=create(name,\${12.().string@d()})	=A1.group(customID)
3	for B2	=12.(null)
4		>A3.run(B3(month(due_date))= amount_payable)
5		>B3.run(~=ifn(~,~[-1]))
6		=A2.record(B2.name B3)

A1: Query 2014 data. A2: Generate a result empty sequence table with 12 months. A3: Group by customID.

A3 ~ B6: Loop the group, B4 sets the payable amount of the corresponding month, B5 sets the blank value to the value of the previous month, and B6 inserts the record into the result sequence table.

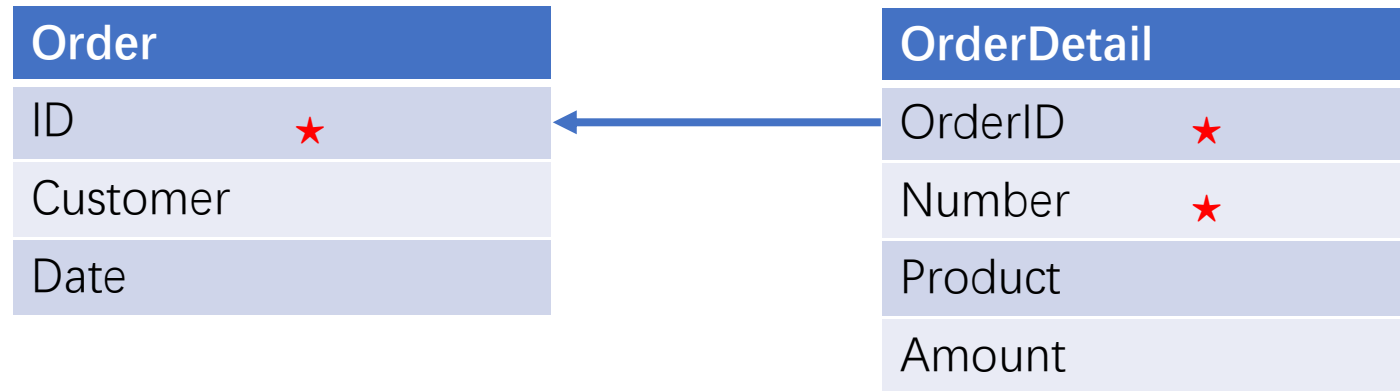
06

Inter-table join column to row conversion



1. Dynamic insertion of sub table into main table

The order table and order detail table are the main sub table relationship. Each order has multiple details, as in the following figure:



The detail data of each order in the order table is variable in length. To find out the following table:

ID	Customer	Date	Product1	Amount1	Product2	Amount2	Product3	Amount3
1	3	20190101	Apple	5	Milk	3	Salt	1
2	5	20190102	Beef	2	Pork	4		
3	2	20190102	Pizza	3				

1. Dynamic insertion of sub table into main table

The SPL script is as follows:

	A	B
1	<code>=connect("db") .query("select * from OrderDetail left join Order on Order.ID=OrderDetail.OrderID")</code>	
2	<code>=B1.group(ID)</code>	<code>=A2.max(~.count()).("Product"+string(~)+", "+"Amount"+string(~)).concat@c()</code>
3	<code>=create(ID, Customer, Date, \${B2})</code>	<code>>A2.run(A3.record([ID, Customer, Date] ~. ([Product, Amount]).conj()))</code>

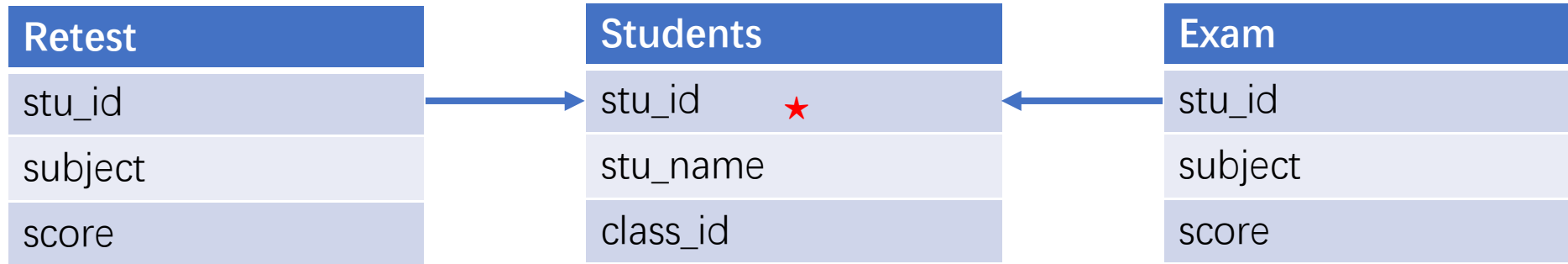
A1: Join the order table and order detail table to retrieve data. A2: Group by order ID.

B2~A3: According to the group with the largest number of members in the group, dynamically generate column names and create sequence table.

B3: Loop the members after grouping, and dynamically put together the data of each group and add them to the sequence table created by A3.

2. Multi table join column to row conversion

There are three tables, namely, student table, exam table and retest table. Stu_id is the join field, as shown in the following figure:



Now we need to query three tables to get the scores of each subject, total score and scores of each student, as shown in the following figure:

stu_id	stu_name	Chinese_score	Math_score	total_score	Chinese_retest	Math_retest
1	Ashley	80	77	156		
2	Rachel	58	67	125	78	
3	Emily	85	56	141		82

2. Multi table join column to row conversion

The SPL script is as follows:

	A	B
1	<code>=connect("db"). query("select t1.stu_id stu_id,t1.stu_name stu_name,t2.subject subject,t2.score score1,t3.score score2 from Students.txt t1 left join Exam.txt t2 on t1.stu_id=t2.stu_id left join Retest.txt t3 on t1.stu_id=t3.stu_id and t2.subject=t3.subject order by t1.stu_id,t2.subject")</code>	
2	<code>=A1.group(stu_id)</code>	<code>=A1.group(subject)</code>
3	<code>=create(stu_id,stu_name,\${(B2.(~.subject+"_score")) "total_score" B2.(~.subject+"_retest")}.string())</code>	
4	<code>>A2.run(A3.record([stu_id,stu_name] B2.(~(A2.#).score1) A2.sum(score1) B2.(~(A2.#).score2)))</code>	

Inter-table join transposition summary



Inter-table join transposition needs to join first

First, the data tables are joined into a single table through association relationship. Next, it's similar to the transpose introduced earlier.

SPL sequence table supports rich functions and can meet most of the operation requirements.

07

Divided by columns



Divided by columns

There is a world urban population table, as shown below:

Continent	Country	City	Population
Africa	Egypt	Cairo	6789479
Asia	China	Shanghai	24240000
Europe	Britain	London	7285000

List the names and population of cities with a population of more than 2 million in Europe and Africa in separate columns (each column is sorted in descending order). The expected result is as follows:

Europe City	Population	Africa City	Population
Moscow	8389200	Cairo	6789479
London	7285000	Kinshasa	5064000
St Petersburg	4694000	Alexandria	3328196

Divided by columns

The idea of column dividing is also to create the target data structure first, and then fill in the data.
The SPL script is as follows:

	A	B
1	<code>=connect("db").query("select * from World where Continent in('Europe', 'Africa') and Population >= 2000000")</code>	
2	<code>=A1.select(Continent:"Europe")</code>	<code>=A1.select(Continent:"Africa")</code>
3	<code>=create('Europe City',Population,'Africa City', Population)</code>	<code>=A3.paste(A2.(City),A2.(Population),B2.(City),B2.(Population))</code>

A1: Connect to the database and retrieve data to select records of more than 2 million people in Europe and Africa.

A2~B2: Take data from Europe and Africa respectively.

A3: Create an empty sequence table according to the target structure. B3: Paste the value sequence directly to the corresponding column using the paste function of the sequence table.



THANKS