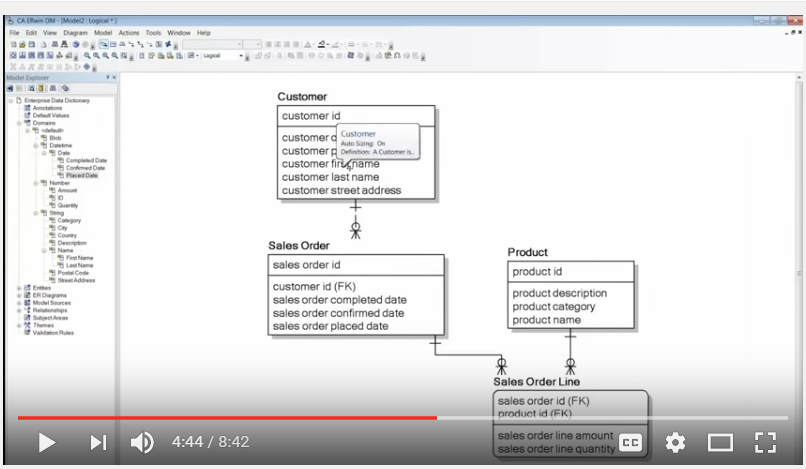
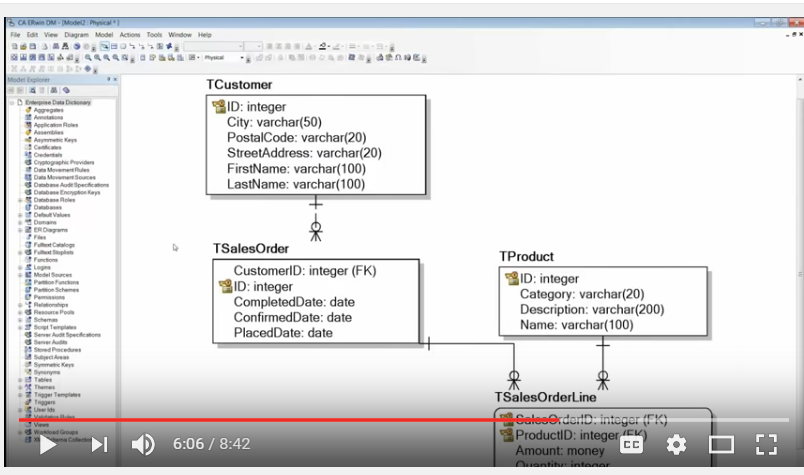
Note we progress from our conceptual model (entity relationships) to our model for the tables the logical view to the how the tables will acutally be built the physical view which as the schema fully specified specified ie types integer varchar etc

SQL PRECEDENCE

Above the logical model



**Above the physical model(view)**

**P**

**ublished on 2011-01-24**

* [Query Syntax](https://www.experts-exchange.com/topics/query-syntax/)

23,821 Points

[6 Comments](https://www.experts-exchange.com/articles/3533/Effective-SQL-Understanding-Order-of-Operations.html#comments)

13,219 Views

6 Endorsements

Last Modified: 2011-01-24

 Awarded

[[](https://www.experts-exchange.com/members/mwvisa1.html)](https://www.experts-exchange.com/members/mwvisa1.html" \t "_blank)

[**Kevin Cross**](https://www.experts-exchange.com/members/mwvisa1.html)

[Follow](https://www.experts-exchange.com/articles/3533/Effective-SQL-Understanding-Order-of-Operations.html)

Introduction

Hopefully the following mnemonic and, ultimately, the acronym it represents is common place to all those reading: **P**lease **E**xcuse **M**y **D**ear **A**unt **S**ally (PEMDAS). Briefly, though, **PEMDAS** is used to signify the [order of operations](http://en.wikipedia.org/wiki/Order_of_operations) being **P**arenthesis (or brackets), **E**xponents (or roots which can be expressed as a fractional exponent 1/n), **M**ultiplication/**D**ivision and **A**ddition/**S**ubtraction.   
  
If you have ever doubted how Math can be used in everyday life, then hopefully this is a good example. I think most forget when dealing with technology or challenges in general, it is all about approaching the situation with a definitive list of known truths and methodologies how to solve issues. You don't have to have the answers going in, just keep it simple and apply what you know like Math rules. SQL syntax is no different and so we will explore in this article how PEMDAS can aid programmers in avoiding some pitfalls or ensuring proper results through being very explicit about the order in which code is evaluated.  
  
If you do this successfully, it should make even complex formulas in your SQL syntax easy to produce.   
  
*So let's get started...*  
  
During the editing of an article some months back, the topic of order of operations came up for discussion and there the data looked as similar to the below:

-------------+---------+---------

name |price |category

-------------+---------+---------

Pen |0.75 | 1

Back Pack |32.50 | 2

Gym Bag |44.25 | 2

Pencil |0.55 | 1

Binder |2.75 | 1

-------------+---------+---------

Select all [Open in new window](https://www.experts-exchange.com/viewCodeSnippet.jsp?codeSnippetId=30-3533-1)

[step=""]**««setup»»**  
*To reproduce the above data, you can execute SQL similar to the below T-SQL example created for MS SQL Server.*  
  
(table structure -- create statement)

create table products(

name varchar(50),

price decimal(8,2),

category int

);

Select all [Open in new window](https://www.experts-exchange.com/viewCodeSnippet.jsp?codeSnippetId=30-3533-2)

(sample data -- insert statement)

insert into products(name, price, category)

values('Pen', 0.75, 1);

insert into products(name, price, category)

values('Back Pack', 32.50, 2);

insert into products(name, price, category)

values('Gym Bag', 44.25, 2);

insert into products(name, price, category)

values('Pencil', 0.55, 1);

insert into products(name, price, category)

values('Binder', 2.75, 1);

Select all [Open in new window](https://www.experts-exchange.com/viewCodeSnippet.jsp?codeSnippetId=30-3533-3)

[/step]  
Now that we have some data, consider the results of this t-sql query:

select name, price, category

from products

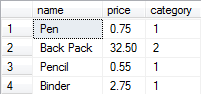
where category = 1

or category = 2

and price <= 40.00

;

Select all [Open in new window](https://www.experts-exchange.com/viewCodeSnippet.jsp?codeSnippetId=30-3533-4)

[](https://filedb.experts-exchange.com/incoming/2010/08_w32/333856/PEMDAS-01-QueryResults1.PNG)  
...and the results of this t-sql query:

select name, price, category

from products

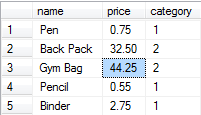
where category = 2

or category = 1

and price <= 40.00

;

Select all [Open in new window](https://www.experts-exchange.com/viewCodeSnippet.jsp?codeSnippetId=30-3533-5)

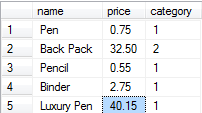
[](https://filedb.experts-exchange.com/incoming/2010/08_w32/333857/PEMDAS-01-QueryResults2.PNG)  
*Why are these different?*  
  
I am guessing the title gives away the answer that this has to do with order of operations with respect to AND and OR operators. Before I address this, it is important to understand which result is correct ... to do so, the original intent must be clear. Clarity is the purpose of parenthesis or brackets, so let's start there.

1. Parenthesis

Using parenthesis aids in maintaining the order in which operators are evaluated in an equation or, in this case, *logical*operators in a conditional. Being in full control of order of operators ensures we achieve the given intent *consistently*.  
  
And just to make this clearer, let's add the following data and re-look at query results 1-1:

insert into products(name, price, category) values('Luxury Pen', 40.15, 1);

Select all [Open in new window](https://www.experts-exchange.com/viewCodeSnippet.jsp?codeSnippetId=30-3533-6)

[](https://filedb.experts-exchange.com/incoming/2010/08_w32/333858/PEMDAS-01-QueryResults1-Rev.PNG)  
If you re-run query 2, you will see the same results because what is happening is that we are getting all records in the first category listed and then only those in the second that are less than or equal to $40.00 in cost.  
  
So let's clarify...  
  
(category is either 1 or 2 with a price <= 40)

(category = 1 or category = 2) and price <= 40.00

Select all [Open in new window](https://www.experts-exchange.com/viewCodeSnippet.jsp?codeSnippetId=30-3533-7)

(category is 1 or category is 2 with a price <= 40)

category = 1 or (category = 2 and price <= 40.00)

Select all [Open in new window](https://www.experts-exchange.com/viewCodeSnippet.jsp?codeSnippetId=30-3533-8)

The addition of the parenthesis around both category filters drastically changes things and makes syntax such as category in (1, 2) also valid, whereas, having category around the last two conditions of the where make it more deliberate that we get results shown in 1-1 illustrations.  
  
A simpler case of this can be seen in the difference between mathematical results of:

select 10 - 2 + 5 from dual;

Select all [Open in new window](https://www.experts-exchange.com/viewCodeSnippet.jsp?codeSnippetId=30-3533-9)

...and...

select 10 - (2 + 5) from dual;

Select all [Open in new window](https://www.experts-exchange.com/viewCodeSnippet.jsp?codeSnippetId=30-3533-10)

**Moral: Be clear. Keep it simple!**

2. Expressions and Element-based Math

**Expressions:**  
"Exponents" is what the ***E***in PEMDAS stands for, but, in the computer world, exponents and roots are achieved through expressions.   
  
For example, the POW or POWER function is used for creating exponents like 10^3 and would be expressed as:

select power(27, 3);

Select all [Open in new window](https://www.experts-exchange.com/viewCodeSnippet.jsp?codeSnippetId=30-3533-11)

In addition, you may have functions like SQRT (square root) or SQUARE as well as the ability of using negative or fractional exponentials with POW[ER] to achieve a given root: 

select power(27, 1/3), power(27, 1.0/3), power(27.0, 1.0/3);

Select all [Open in new window](https://www.experts-exchange.com/viewCodeSnippet.jsp?codeSnippetId=30-3533-12)

Note: In MS SQL Server, the data type of the numbers involved in the POWER function matter. Therefore, the three expressions above yield three different results in Transact-SQL.  
  
At the risk of veering too far off tangent, absolute value (e.g., |x|) is represented using expressions also. The correlation I like to think of is the ^ which can sometimes be used by us mathematical types to represent exponents (when superscript numbers are not available) and the | are both bitwise operators, which we will talk about later; therefore, the ^ cannot be used as the exponent operator.  
  
Since functions contain parenthesis, you can consider that the operator precedence is maintained here as you evaluate everything inside the brackets before applying the formula of the expression itself.   
  
This can be seen in practical use in my recent article on [Conditional Aggregates](http://e-e.com/A_3527.html). As exemplified by the percentage example query, the expressions will evaluate first so that later operators like multiplication and division have a value upon which to act.  
  
  
**Moral: Know your expressions and be creative in how you use them!**  
  
MSACC: <http://www.techonthenet.com/access/functions/>  
MSSQL: <http://msdn.microsoft.com/en-us/library/ms177516.aspx>  
MySQL: <http://dev.mysql.com/doc/refman/5.5/en/mathematical-functions.html>  
PLSQL: <http://www.techonthenet.com/oracle/functions/>  
  
  
**Element-based Math:**  
Another "E" (as I am taking liberty of calling it element-based math) is for the priority given to unary operations: involving operators that act on a single *element*. Check your SQL systems implementation for specific unary operators available, but examples are: ~ (bitwise NOT), - (negative) and + (positive).

3. Multiplication (or Modulo) and Division then Addition and Subtraction

**Multiplication/Modulo/Division:**  
These operators should be pretty self explanatory (except maybe [Modulo](http://en.wikipedia.org/wiki/Modulo_operation)), but as seen above when the proper order of operations is not considered, the results could be unexpected or inconsistent with the business intent. This is where it is best to be very explicit and possibly over utilize parenthesis in your syntax.  
  
Subsequently, beyond assuring proper order of operations is clear, it is more efficient to simplify your mathematical formulas when possible thus eliminating multiple operations, potentially, to begin with.  
  
For example, you may see an equation like the following where an interest rate stored as a whole number (e.g., 10%) is divided by and then multiplied by a principal column and a literal value (e.g., 5 for number of years):

select interest\_rate / 100 \* principal \* 5

from loans;

Select all [Open in new window](https://www.experts-exchange.com/viewCodeSnippet.jsp?codeSnippetId=30-3533-13)

Reducing this formula to the below saves on extra processing by SQL:

select interest\_rate \* principal \* 0.05 ...

Select all [Open in new window](https://www.experts-exchange.com/viewCodeSnippet.jsp?codeSnippetId=30-3533-14)

Additionally, as eluded to in the [Conditional Aggregates](http://e-e.com/A_3527.html) article, this simplification of operations has just helped us avoid side effects of implicit type conversion to INT here when evaluating 5 / 100 which would normally come out to be 0 in MS SQL.   
  
  
**Addition/Subtraction:**  
As above, this is straight-forward, but just remember its order is after the above arithmetic operators if no parenthesis are used and you should try to simplify when possible so that you don't have literal values acting as operands when you can do the math offline and plug in the product, quotient, sum or difference directly into the query.  
  
  
**Moral: Do the math; check your answer!**

BONUS: Type Conversions

Consequently, it is good to note at this point, while on this train of thought (i.e., dealing with arithmetic operators), that you should take into consideration the type conversions and the order in which they occur naturally to ensure you understand their effect on your equation's results.  
  
For example, note the difference in results of the same division problem from above, using explicit conversion:

convert(decimal(10,4), 5 / 100)

Select all [Open in new window](https://www.experts-exchange.com/viewCodeSnippet.jsp?codeSnippetId=30-3533-15)

convert(decimal(10,4), 5) / 100

Select all [Open in new window](https://www.experts-exchange.com/viewCodeSnippet.jsp?codeSnippetId=30-3533-16)

Going back to the Expressions section, the latter works since the CONVERT expression will occur first then the division by 100, changing this evaluation from the division of two Integers to normal decimal math (or from a different point of view forces an implicit conversion of the literal 100 in denominator to match the numerator). Note well that often doing these explicit conversions of operands can increase overall query performance.

BONUS: Bitwise and Logical Operations

**Bitwise Operations:**  
Circling back to the precedence order of addition/subtraction, the [bitwise operators](http://en.wikipedia.org/wiki/Bitwise_operation) AND (&), OR (|) and Exclusive OR (^ or XOR) fall on equal plain; therefore, the importance is these operations will occur after multiplication/division but before logical operations. This fact ensures that you can do simple things like:

select name, price, category

from products

where price > category^2;

Select all [Open in new window](https://www.experts-exchange.com/viewCodeSnippet.jsp?codeSnippetId=30-3533-17)

Keeping in mind ^2 could have been any other operation above like \*2 or +2.  
  
**Logical Operations:**

2b OR NOT 2b

Select all [Open in new window](https://www.experts-exchange.com/viewCodeSnippet.jsp?codeSnippetId=30-3533-18)

*...that is the question!*  
  
The NOT will be evaluated first here and then the connecting logical operator like the OR shown here. Following this, the AND operator has precedence and then the OR and other comparative statements similar to it (e.g., LIKE, IN, BETWEEN, ANY, ALL, SOME, etc.  -- check your SQL implementation for a more complete/correct list in your system).   
  
One way to think of this is AND limits a condition or the number of records further as you are potentially dividing your already filtered recordset into smaller groups: ones that meet the additional criteria and the ones that don't. Along this same line of reasoning, OR can be likened to adding dissimilar rows to the same resultset. The key difference there is this doesn't equate to a straight addition of row counts as a single record may satisfy both conditions, so don't waste too much time trying to figure out why a < 5 which results in 10 rows combined with OR a < 3 which results in 5 rows doesn't equal 15 rows.    
  
*Eureka!*  
  
Now we understand the answer to our introductory question, right?

category = 2 or category = 1 and price <= 40.00

Select all [Open in new window](https://www.experts-exchange.com/viewCodeSnippet.jsp?codeSnippetId=30-3533-19)

Knowing what we know now, it should be clear that the above doesn't translate to *category of 2 or 1 and*... but instead:

category = 2 or (category = 1 and price <= 40.00)

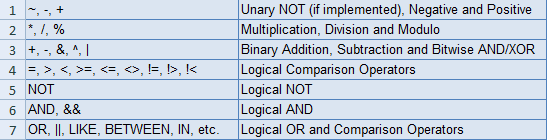
Select all [Open in new window](https://www.experts-exchange.com/viewCodeSnippet.jsp?codeSnippetId=30-3533-20)

Conclusion

In summary, know your order of operations and utilize them in your everyday SQL to code more efficiently and effectively.  
  
**Let's recap:**

 Use parenthesis - a lot of the pitfalls folks get into with data calculations using SQL are due to a lack of explicit consideration for operator precedence and so later when code is innocently reorganized by another developer, the results can be drastically changed as we saw in flipping which category appeared first in our where condition in the introduction. With parenthesis, someone maintaining the original code would know for certain the intent and as such are likelier to not make such a change without consideration for its impact on the meaning of the query.

 Expressions are evaluated next, so use this to your advantage for things like type conversion.

 Keep the simple math and logical conditions just that: simple! As a reminder of the unary, binary and logical operators’ precedence, see the illustration below: [](https://filedb.experts-exchange.com/incoming/2010/08_w33/335767/OrderOfOperatorsChart.PNG)

Hope that helps...  
  
*Thanks as usual for reading!*  
  
Best regards and happy coding,  
  
Kevin C. Cross, Sr. ([mwvisa1](http://www.experts-exchange.com/M_2758051.html))  
  
=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=  
**If you found this article helpful**, please click the **Yes** button after the question just below. This will give me (the author) a few points and might encourage me to write more articles.   
  
If you didn't or otherwise feel the need to vote No, please first leave a comment to give me a chance to answer and perhaps to improve this article.   
  
***Thank you!***  
=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=

**6**

[Comment](https://www.experts-exchange.com/articles/3533/Effective-SQL-Understanding-Order-of-Operations.html#writeComment)

Author:[Kevin Cross](https://www.experts-exchange.com/members/mwvisa1.html)

[[](https://www.experts-exchange.com/members/mwvisa1.html)](https://www.experts-exchange.com/members/mwvisa1.html" \t "_blank)

* 

4

* 

* 

6 Comments

[[](https://www.experts-exchange.com/members/mwvisa1.html)](https://www.experts-exchange.com/members/mwvisa1.html)

[LVL 59](https://www.experts-exchange.com/members/mwvisa1.html)

Active 4 days ago

**Author Comment**

by:[Kevin Cross](https://www.experts-exchange.com/members/mwvisa1.html)2010-08-11

Thank you again to all for stopping by and reading this, but I wanted to extend a ***special thank you***to EE members [tedbilly](http://www.experts-exchange.com/M_1204358.html" \t "_blank) and [DanRollins](http://www.experts-exchange.com/M_330221.html" \t "_blank). Their commentary during authoring of this article was very instrumental in how it is currently shaped; therefore, I for one am greatly appreciative and hope that our combined efforts were worth it for all you readers out there.   
  
While I have your attention again and we are on this subject ...  
  
**More on Bitwise vs Logical Operators:**  
A number of systems like MS SQL use AND (logical) and & (bitwise) operators; therefore, there tends to not be as much confusion on which is which or that the latter is a binary operator like plus and thus is doing a specific operation on the operands involved and not just a simple comparison of both conditions being true as the former does.  
  
However, note that in some database systems like MySQL, && is used to represent the logical AND, making these look to be doing the same thing:

if (1 & 0) ...

Select all [Open in new window](https://www.experts-exchange.com/viewCodeSnippet.jsp?codeSnippetId=90-18040-1)

if (1 && 0) ...

Select all [Open in new window](https://www.experts-exchange.com/viewCodeSnippet.jsp?codeSnippetId=90-18040-2)

Both of the above end up in the false or else case of the if statement, but that is because 0 is bit value of false; 1, (typically) true. The first code is doing bit by bit comparison (using logical AND) of the two binary values 1 and 0, resulting in this case as 0. The second is logically comparing if the first operand evaluates to true AND the second also evaluates to true. Type conversion aids in the second by implicitly converting 1 and 0 to their boolean counterparts and then does the logical comparison.  
  
Consequently, the same is true with | (bitwise) and || (logical) OR operators.  
  
This distinction should help explain why this code can result in a null pointer exception:

if ((a != null) & (a.equals(b)))

Select all [Open in new window](https://www.experts-exchange.com/viewCodeSnippet.jsp?codeSnippetId=90-18040-3)

Where as this one is protected:

if ((a != null) && (a.equals(b)))

Select all [Open in new window](https://www.experts-exchange.com/viewCodeSnippet.jsp?codeSnippetId=90-18040-4)

See with a binary operation, you have to evaluate both sides of the operator first and then you perform the operation upon them. As a logical AND, contrarily, being that both conditions MUST be true for the statement to be true, once the first condition equates to false (i.e., a == null) then further processing stops avoiding the null pointer exception caused by trying to use the null value as the object its variable represents. This is the same with OR since your first statement would check the inverse (a == null) which equating to true makes the rest of the statement irrelevant since ONLY one condition has to be true in that case.  
  
Okay, think I am starting to ramble, so best regards and happy coding until the next adventure!  
  
Kevin

[[](https://www.experts-exchange.com/members/mwvisa1.html)](https://www.experts-exchange.com/members/mwvisa1.html)

[LVL 59](https://www.experts-exchange.com/members/mwvisa1.html)

Active 4 days ago

**Author Comment**

by:[Kevin Cross](https://www.experts-exchange.com/members/mwvisa1.html)2010-08-13

I was helping someone today with SQLite (<http://www.sqlite.org/lang_expr.html>) which I had not used before and I was reminded of always using parenthesis even if they don't appear to be needed as it turns out that SQLite puts || (concatenation) operator as of higher precedence than the arithmetic operators; therefore, x\*y || ' units' doesn't work out to be '{product} units' but instead NULL or not evaluated as you end up with x \* 'y units' which can't compute product of a string and number.   
  
Now we already knew that this is slightly different based on system (programming language), so this is probably not earth-shattering there is a platform with different rules, but thought I would mention for those SQLite readers who do venture here and, most importantly, because it had a bit on COLLATION being top of the precedence ladder that reminded me of a good point to mention here.   
  
COLLATION, probably along the nature of the Type Conversions section, will be applied to the string / column before comparison thus results may be different using implicit versus explicit.   
  
In other words -- the following are not equivalent (example from MS SQL 2008):

N'A' COLLATE SQL\_Latin1\_General\_Cp1\_CS\_AS = N'a'

Select all [Open in new window](https://www.experts-exchange.com/viewCodeSnippet.jsp?codeSnippetId=90-18176-1)

N'A' COLLATE SQL\_Latin1\_General\_Cp1\_CI\_AS = N'a'

Select all [Open in new window](https://www.experts-exchange.com/viewCodeSnippet.jsp?codeSnippetId=90-18176-2)

Depending on what your server's collation is set to, those may also differ from this:

N'A' = N'a'

Select all [Open in new window](https://www.experts-exchange.com/viewCodeSnippet.jsp?codeSnippetId=90-18176-3)

If you can imagine, instead of simple capital and lowercase comparison that we are dealing accents for example. I could probably go on, so if you are interested -- you should probably play around with some of the different collations out there like binary and see how they change your sort order in a simple ORDER BY.  
  
Enjoy!  
  
Kevin

[[](https://www.experts-exchange.com/members/mwvisa1.html)](https://www.experts-exchange.com/members/mwvisa1.html)

[LVL 59](https://www.experts-exchange.com/members/mwvisa1.html)

Active 4 days ago

**Author Comment**

by:[Kevin Cross](https://www.experts-exchange.com/members/mwvisa1.html)2011-01-23

I recently attended and presented at my first [SQLSaturday!](http://www.sqlsaturday.com/" \t "_blank) this past weekend and one of my presentations was on this very subject.  To add some value to the presentation, I included some reminders on the order in which the portions of a SQL SELECT statement are evaluated in those slides that I have cleaned up a bit and uploaded here:    
 [20110122-SQLSat45-OrderOperation.pptx](https://filedb.experts-exchange.com/incoming/2011/01_w05/400099/20110122-SQLSat45-OrderOperation.pptx)  
  
Since the demos are compiled of the SQL queries showed here between the Article body and comments previous to this one, I have not attached the 3 DEMO SQL files here; however, they will be available as promised via the [SQLSaturday.com](http://sqlsaturday.com/) web-site.   
  
**Special thanks to all those of you that came out to support me in Louisville!**  
  
Kevin

[[](https://www.experts-exchange.com/members/hongjun.html)](https://www.experts-exchange.com/members/hongjun.html)

[LVL 33](https://www.experts-exchange.com/members/hongjun.html)

**Expert Comment**

by:[hongjun](https://www.experts-exchange.com/members/hongjun.html)2011-01-24

I remember it as BODMAS  
  
(B)rackets  
  
(O)rder  
  
(D)ivision  
  
(M)ultiplication  
  
(A)ddition  
  
(S)ubtraction  
  
But PEMDAS is interesting enough.

[[](https://www.experts-exchange.com/members/mwvisa1.html)](https://www.experts-exchange.com/members/mwvisa1.html)

[LVL 59](https://www.experts-exchange.com/members/mwvisa1.html)

Active 4 days ago

**Author Comment**

by:[Kevin Cross](https://www.experts-exchange.com/members/mwvisa1.html)2011-01-24

Yes, true.  I guess I am trained to think Exponents instead of Orders | Indices as well as my use of parenthesis, but BODMAS or BIDMAS works just fine! **\*smile\***  
  
***Thanks for reading!***

[[](https://www.experts-exchange.com/members/SimonAdept.html)](https://www.experts-exchange.com/members/SimonAdept.html)

[LVL 18](https://www.experts-exchange.com/members/SimonAdept.html)

**Expert Comment**

by:[SimonAdept](https://www.experts-exchange.com/members/SimonAdept.html)2015-01-08

Thanks Kevin. Good article. I think that when I went to school they weren't much into mnemonics. I like both PEMDAS and BODMAS, but the precedence is just second nature. I very rarely get it wrong, but I don't consciously think about it.