ΠΡΟΓΡΑΜΜΑ ΜΕΤΑΠΤΥΧΙΑΚΩΝ ΣΠΟΥΔΩΝ

"ΠΛΗΡΟΦΟΡΙΚΗ" 4ος κύκλος / Α' ΕΞΑΜΗΝΟ

ΜΑΘΗΜΑ ΕΠΙΛΟΓΗΣ ΛΟΓΙΚΟΣ ΠΡΟΓΡΑΜΜΑΤΙΣΜΟΣ

Μάρτιος 2022

Sudoku σε Prolog

Ον. Επώνυμο: **Κωνσταντίνος Κολιός**

Αρ. Μητρώου: **ΜΠΠΛ 21032**

Περιεχόμενα

Εισαγωγή.....3
Τί είναι η prolog;.....3
Διαφορά διαδικαστικών γλωσσών και Prolog..... 3
Γεγονότα και Κανόνες στην Prolog..... 3
Σκόπος της εργασίας..... 4
Πώς παίζεται το sudoku;..... 5
Υλοποίηση σε Prolog..... 6
Πρώτος τρόπος:.....11
Δεύτερος τρόπος:..... 12
Κώδικας.....23

Εισαγωγή

Τί είναι η prolog;

Η prolog είναι μια συμβολική γλώσσα προγραμματισμού που βασίζεται στην κατηγορηματική λογική. Αναπτύχθηκε στις αρχές του 1970 από τους Robert Kowalski και Alain Colmerauer. Η πρώτη υλοποίηση οφείλεται στην γαλλική ερευνητική ομάδα του Alain Colmerauer στο πανεπιστήμιο Luminy της Μασσαλίας, η οποία ανέπτυξε ένα πρόγραμμα απόδειξης θεωρήματων για επεξεργασία φυσικής γλώσσας.

Διαφόρα διαδικαστικών γλωσσών και Prolog.

Σε ένα οποιοδήποτε πρόγραμμα, διακρίνουμε το τμήμα της λογικής και το τμήμα του ελέγχου. Σύμφωνα με τον Κοβάσκι : πρόγραμμα = λογική + έλεγχος.

Στο διαδικαστικό προγραμματισμό, το τμήμα της λογικής και το τμήμα του ελεγχου έιναι αλληλένδετα και δεν διαχωρίζονται δηλαδή ο προγραμματιστής πρέπει να καθορίσει επακριβώς τη ροή ελέγχου του προγράμματος ανάλογα με τη λογική και τα διαθέσιμα δεδομένα του προβλήματος.

Στην prolog γίνεται διαχωρισμός λογικής - ελέγχου δηλαδή χρείαζεται να περιγραφεί μόνο η λογική του προς επίλυση προβλήματος ενώ ο έλεγχος αφήνεται στο σύστημα.

Γεγονότα και Κανόνες στην Prolog.

Η λογική ενός προγράμματος στην Prolog είναι ένα σύνολο προτάσεων που περιγράφει τα δεδομένα του προβλήματος και τις σχέσεις που τα συνδέουν.

Οι προτάσεις αυτές λέγονται προτάσεις τύπου Horn (Horn clauses) και αποτελούν υποσύνολο της κατηγορηματικής λογικής πρώτης τάξης.

Υπάρχουν δύο είδη προτάσεων, τα γεγονονότα και οι κανόνες. Τα γεγονότα εκφράζουν σχέσεις ανάμεσα στα αντικείμενα και αποτελούν τα δεδομένα του προβλήματος. Αποτελούν την πιο απλή μορφή γνώσης, και μοιάζουν με τα records των βάσεων δεδομένων.

Για παράδειγμα αν θέλουμε να δηλώσουμε ότι "ο Κωνσταντίνος είναι ο πατέρας της Σοφίας" το γράφουμε με την εξής μορφή:

```
father(konstantinos,sofia).
```

Οι κανόνες εκφράζουν γενικότερες σχέσεις ανάμεσα στα αντικείμενα, οι οποίες ορίζονται με τη βοήθεια άλλων σχέσεων. Για παράδειγμα η σχέση γονιού ορίζεται με δύο κανόνες:

"ο Χ είναι γονιός του Υ εάν ο Χ είναι πατέρας του Υ"
"Η Χ είναι γονιός του Υ εάν η Χ είναι μητέρα του Υ"

Σε Prolog το γράφουμε:

```
parent(X,Y) :- father(X,Y).
parent(X,Y) :- mother(X,Y).
```

Όπου Χ,Υ είναι μεταβλήτες και το σύμβολο ':-' αντιπροσωπεύει το λογικό εάν.

Σκόπος της εργασίας

Στην συγκεκριμένη εργασία θα παρουσιαστεί μια υλοποίηση που θα μπορεί ο χρήστης να λύσει άμεσα οποιοδήποτε συνδυασμό επιθυμεί για το παιχνίδι sudoku 9x9. Παραλλήλα θα υπάρχει πλήρης καταγραφή όσων κανόνων θα χρησιμοποιηθούν και θα εκτελεστούν σχετικά παραδείγματα.

Πώς παίζεται το sudoku;

Το sudoku είναι παζλ που βασίζεται στην λογική. Στόχος είναι να συμπληρωθούν όλα τα κουτάκια σε ένα πίνανα έστω 9x9 διαστάσεων ώστε κάθε στήλη και κάθε σειρά και κάθε κουτάκι 3x3 να περιέχουν όλα τα ψηφία απο το 1 μέχρι το 9, οπότε αν υπάρχουν ήδη συμπληρωμένα κουτάκια θα υπάρξει μόνο μια δυνατή λύση.

Ένα απλό παραδείγμα κατανόησης του παραπάνω ορισμού είναι το εξής :

Πρόβλημα:

5	3			7				
6			1	9	5			
	9	8					6	
8				6				3
4			8		3			1
7				2				6
	6					2	8	
			4	1	9			5
				8			7	9

Λύση:

5	3	4	6	7	8	9	1	2
6	7	2	1	9	5	3	4	8
1	9	8	3	4	2	5	6	7
8	5	9	7	6	1	4	2	3
4	2	6	8	5	3	7	9	1
7	1	3	9	2	4	8	5	6
9	6	1	5	3	7	2	8	4
2	8	7	4	1	9	6	3	5
3	4	5	2	8	6	1	7	9

Δηλαδή δεν πρέπει κάποιος αριθμός να υπάρχει 2 φορες στην ιδια γραμμή, την ίδια στήλη, στο ίδιο 3x3 τετράγωνο.

Υλοποίηση σε Prolog

Απαραίτητη προυπόθεση για να δοκιμάσουμε τον κώδικα μας είναι να έχουμε εγκαταστήσει την Prolog στον υπολογιστή μας. Περισσότερες πληροφορίες για το πώς θα το κάνετε μπορείτε να βρείτε στο αντίστοιχο link:

https://www.geeksforgeeks.org/how-to-install-swi-prolog-on-windows/

Έπειτα ανοίγουμε ένα ASCII αρχείο '.txt' και το αποθηκεύουμε με την κατάληξη '.pl' ώστε να μπορούμε να το ενσωματώσουμε αργότερα στον Editor που κατεβάσαμε από το παραπάνω βήμα. (Αποθηκεύουμε δηλάδη το αρχείο ώς πχ. 'Sudoku.pl')

Μπορούμε να χρησιμοποιήσουμε είτε τα σύμβολα "/* */" για να γράφουμε σχόλια στον κώδικά μας για να μας βοηθήσει στην καλύτερη ανάγνωση της κάθε λειτουργίας που

χρησιμοποιούμε. Επίσης αν θέλουμε να κάνουμε σχόλιο μόνο για μια γραμμή μπορούμε να χρησιμοποιήσουμε το σύμβολο “ % ”.

Αρχικά στην υλοποίηση θα χρειάστουμε την βιβλιοθήκη clpfd. Οπότε θα ξεκινήσουμε τον κώδικά μας γράφοντας:

```
:- use_module(library(clpfd)).
```

Αξίζει να σημειωθεί, ότι όταν τελειώνουμε την κάθε γραμμή κώδικα που χρησιμοποιούμε θα πρέπει να βάζουμε και το σύμβολο “.” και αν θέλουμε να γράψουμε το λογικό και (“and”) αυτό γίνεται με το σύμβολο “,”, ενώ για το λογικό ‘ή’ (or), χρησιμοποιούμε το σύμβολο ‘;’.

Στην συνέχεια ξεκινάμε να γράψουμε τον πρώτο μας κανόνα:

```
sudoku(Rows) :-
     length(Rows,9),
     maplist(same_length(Rows),Rows),
     append(Rows, Vs), Vs ins 1..9,
     maplist(all_distinct,Rows),
     transpose(Rows,Columns),
     maplist(all_distinct,Columns).
```

Οι γραμμή κώδικα :

```
     length(Rows,9),
```

σημαίνει το μήκος της γραμμής (Rows) πρέπει να είναι τουλάχιστον μέχρι το 9.

Επειδή κάθε γραμμή πρέπει να έχει το ίδιο μήκος με κάθε γραμμή της λίστας θα πρέπει να χρησιμοποιήσουμε:

```
maplist(same_length(Rows),Rows),
```

Υπάρχει ήδη μια λειτουργία στην prolog που λέγεται maplist και χρησιμοποιείται έαν ο κανόνας χρησιμοποιηθεί σε όλα τα στοιχεία της λιστας, όπως υπάρχει και η λειτουργία same_length που στην ουσία βλέπει αν δύο λίστες A, B, έχουν ίδιο μήκος.

Έπειτα θα χρειαστούμε δύο λειτουργίες που η πρώτη θα ενώνει δύο λίστες και η δεύτερη θα ορίζει τα στοιχεία να παίρνουν τιμές απο το 1 μέχρι το 9. Αυτό μπορούμε να το επιτύχουμε στην Prolog με την χρήση της append (για την συνένωση των λιστών) και με την χρήση της ins 1..9 (για να πάρει τις τιμές απο το 1 μεχρι το 9). Οπότε γράφουμε:

```
append(Rows, Vs), Vs ins 1..9,
```

Ακόμα θα πρέπει να ορίσουμε οτι οι παραπάνω αριθμοί θα πρέπει να υπάρχουν μόνο μια φορά στην ίδια γραμμή. Αυτό θα το επιτύχουμε χρησιμοποιώντας μια έτοιμη λειτουργία στην Prolog που ονομάζεται all_distinct που απαιτεί όλα τα στοιχεία μιας λίστας να έχουν διαφορετική τιμή. Οπότε γράφουμε:

```
maplist(all_distinct,Rows),
```

Θα πρέπει τώρα να κάνουμε τις γραμμές να γίνοντα αντίστοιχα στήλες και οι στήλες αυτές να είναι γραμμές και παραλλήλα να κάνουμε τον παραπάνω έλεγχο όπως και στις γραμμές ώστε κάθε στήλη να μην έχει μια τιμή παράπανω από μια φορά. Αυτο θα γίνει με την χρησιμοποίηση της έτοιμης λειτουργίας transpose που στην ουσία μετατρέπει την κάθε γραμμή σε στήλη.

```
transpose(Rows,Columns),
```

Και ο έλεγχος για τις στείλες θα γίνει:

```
maplist(all_distinct,Columns).
```

Για ευκολία στο παράδειγμα μας θα χρειαστεί να ονομάσουμε την κάθε γραμμή διαφορετικά και να χωρίσουμε την κάθε γραμμή σε κουτάκια των τριών, το οποίο πολύ απλά θα το γράψουμε:

```
Rows = [As,Bs,Cs,Ds,Es,Fs,Gs,Hs,Is],
blocks(As, Bs, Cs),
blocks(Ds, Es, Fs),
blocks(Gs, Hs, Is).
```

Τέλος αυτό που απομένει να ορίσουμε είναι τον κανόνα για το κάθε τετράγωνο που ονομάζουμε blocks.

```
blocks( [], [], []).
blocks([N1,N2,N3|Ns1],[N4,N5,N6|Ns2],[N7,N8,N9|Ns3]):-
     all_distinct([N1,N2,N3,N4,N5,N6,N7,N8,N9]),
     blocks(Ns1, Ns2, Ns3).
```

Αρχικά θα ορίσουμε να δέχεται 3 λίστες σαν όρισμα οπότε θα γράψουμε :

```
blocks( [], [], []).
```

Θα πρέπει παράλληλα να ορίσουμε οτι κάθε λίστα ξεκινάει με 3 παραμέτρους δηλαδή 1 2 3,4 5 6,7 8 9 για να φτιάξουμε το κουτί 3x3. Αυτό στην Prolog θα το γράψουμε με την εξής μορφή:

```
blocks([N1,N2,N3|Ns1],[N4,N5,N6|Ns2],[N7,N8,N9|Ns3]):-
```

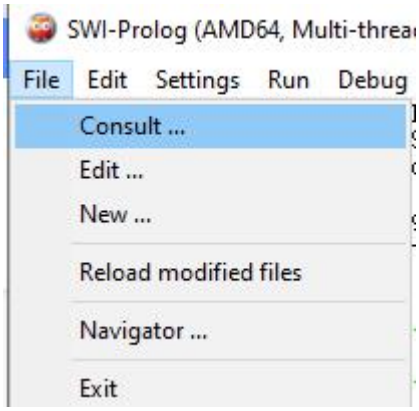
Και θα πρέπει να χρησιμοποιήσουμε την έτοιμη λειτουργία all_distinct για να ορίσουμε οτι το κάθε κουτάκι δεν μπορεί να έχει την ίδια τιμή παραπάνω απο μία φορά οπότε:

```
all_distinct([N1,N2,N3,N4,N5,N6,N7,N8,N9]),
blocks(Ns1, Ns2, Ns3).
```

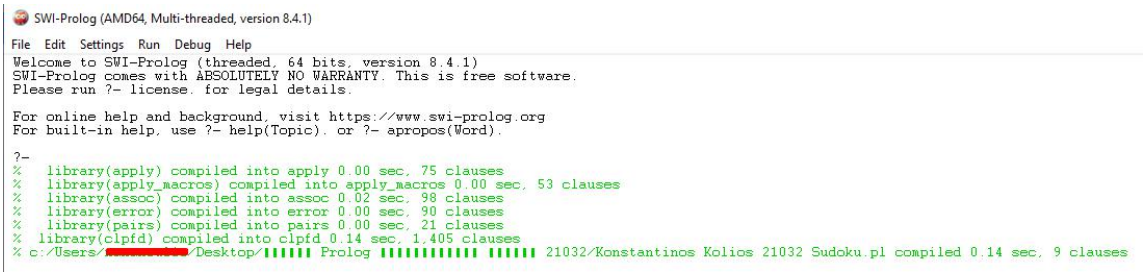
Και πλέον είμαστε έτοιμοι να δοκιμάσουμε τον κώδικα μας. Συνολικά ο κώδικας του πρόγραμματος που θα χρησιμοποίησουμε είναι:

```
/* For this solution we need to use the library clpfd */
/* For more documentation about this library please use this link -> https://www.swi-prolog.org/man/clpfd.html */
:- use_module(library(clpfd)).

sudoku(Rows) :-
        % Rows must be at least length of 9
        length(Rows, 9),
        % Each of a row must has the same length as the list rows so thats why we use same_length.
        maplist(same_length(Rows), Rows),
        % The concatenation of all elements of the list is an intenger between in range [1,9].
        append(Rows, Vs), Vs ins 1..9,
        % We use all_distinct with maplist to defind that every row can't have the same number in above range twice.
        maplist(all_distinct, Rows),
        % transpose turns the rows into columns so the columns are now rows.
        transpose(Rows, Columns),
        % so we use maplist with all distinct now for the columns.
        maplist(all_distinct, Columns),
        % Now we give names in each rows.
        Rows = [As,Bs,Cs,Ds,Es,Fs,Gs,Hs,Is],
        % and we split each block by three elements
        blocks(As, Bs, Cs),
        blocks(Ds, Es, Fs),
        blocks(Gs, Hs, Is).

/* We need to define the usage of blocks &       */
/* We need to pass three Empty list's []         */
/* The reason of using block is to ensure each block 3x3 to contain distinct intengers in every combination in range [ ins 1..9]  */
/* Example [1,2,3,4,5,6,7,8,9] == TRUE;          */
/* Example [1,2,3,3,5,6,7,8,9] == FALSE;         */
/* Example [9,1,2,4,5,6,8,7,3] == TRUE;          */
blocks([], [], []).
blocks([N1,N2,N3|Ns1], [N4,N5,N6|Ns2], [N7,N8,N9|Ns3]) :-
        all_distinct([N1,N2,N3,N4,N5,N6,N7,N8,N9]),
        blocks(Ns1, Ns2, Ns3).
```

Αρχικά θα ανοίξουμε τον editor SWI-Prolog και θα πατήσουμε την ένδειξη Consult που θα βρούμε απο το path File > Consult όπως στην παρακάτω εικόνα:



Έπειτα διαλέγουμε το αρχείο που θέλουμε να τρέξουμε στην Prolog (Προσοχή!! πρέπει να έχει την κατάληξη .pl) και αν δεν υπάρχει κάποιο συντακτικό πρόβλημα στο κώδικά μας θα μοιάζει με την εξής μορφή:



Όπότε τώρα βρισκόμαστε στην κατάλληλη θέση να δοκιμάσουμε την δύναμη της Prolog.

Πρώτος τρόπος:

Αν θέλουμε η πρώτη γραμμή μας να ξεκινάει πχ. Απο 9 θα το δοκιμάσουμε γράφοντας την εντολή:

```
?- Rows = [[9|_]|_],
     sudoku(Rows),
     maplist(label,Rows),
     maplist(portray_clause,Rows).
```

Αποτέλεσμα οθόνης:

```
?- Rows = [[9|_]|_],sudoku(Rows),
|    maplist(label,Rows),maplist(portray_clause,Rows).
[9, 1, 2, 3, 4, 5, 6, 7, 8].
[3, 4, 5, 6, 7, 8, 1, 2, 9].
[6, 7, 8, 1, 2, 9, 3, 4, 5].
[1, 2, 3, 4, 5, 6, 8, 9, 7].
[4, 5, 6, 8, 9, 7, 2, 1, 3].
[7, 8, 9, 2, 1, 3, 4, 5, 6].
[2, 3, 7, 5, 6, 1, 9, 8, 4].
[5, 6, 1, 9, 8, 4, 7, 3, 2].
[8, 9, 4, 7, 3, 2, 5, 6, 1].
Rows = [[9, 1, 2, 3, 4, 5, 6, 7|...], [3, 4, 5, 6, 7, 8, 1|...], [6, 7, 8, 1, 2, 9
|...], [1, 2, 3, 4, 5|...], [4, 5, 6, 8|...], [7, 8, 9|...], [2, 3|...], [5|...],
[...|...]]
```

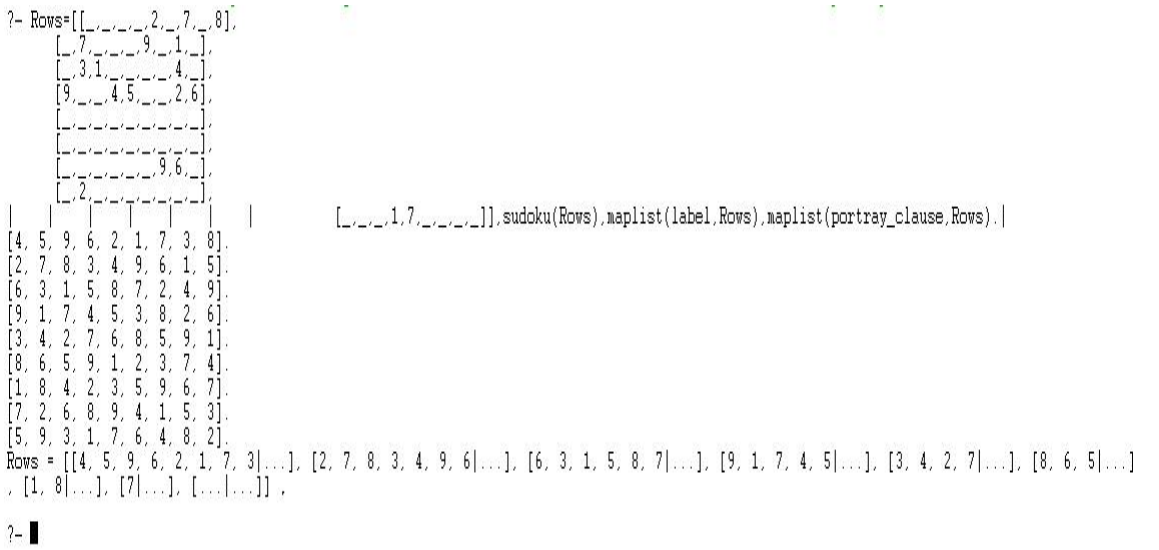
Η παραπάνω εντολή χρησιμοποίει δύο λειτουργίες που δεν αναφέραμε παραπάνω.Η πρωτή λειτουργία label υπάρχει ήδη στην βιβλιόθηκη που χρησιμοποιήσαμε αρχικά στο πρόγραμμα μας(clpfd) και χρησιμοποιείτε για την συστηματική δοκιμή τιμών για τις μεταβλητές πεπερασμένου τομέα μέχρι να προσπελαστούν ολές. Η εντολή portray_clause είναι και αυτή μια έτοιμη λειτουργία που χρησιμοποιείτε για να εκτυπώνει “όμορφα” το αποτέλεσμα μας.

Δεύτερος τρόπος:

Μπορούμε να βάζουμε ένα sudoku 9x9 βάζοντας κάθε γραμμή ή στήλη όποιον αριθμό επιθυμούμε και θα έχουμε την αντίστοιχη απάντηση!

```
?- Rows=  [[_,_,_,_,2,_,7,_,8],
           [_,7,_,_,_,9,_,1,_],
           [_,3,1,_,_,_,_,4,_],
           [9,_,_,4,5,_,_,2,6],
           [_,_,_,_,_,_,_,_,_],
           [_,_,_,_,_,_,_,_,_],
           [_,_,_,_,_,_,9,6,_],
           [_,2,_,_,_,_,_,_,_],
           [_,_,_,1,7,_,_,_,_]],
      sudoku(Rows),
      maplist(label,Rows),
      maplist(portray_clause,Rows).
```

Αποτέλεσμα οθόνης:



Σας παραθέτω και τον κώδικα απο αρχείο sudoku.pl για να δοκιμάσετε τους δικούς σας συνδυασμούς!

Κώδικας:

```
/* For this solution we need to use the library clpfd */
/* For more documentation about this library please use this
link -> https://www.swi-prolog.org/man/clpfd.html */

:- use_module(library(clpfd)).

sudoku(Rows) :-
    % Rows must be at least length of 9
         length(Rows, 9),
     /* Each of a row must has the same length as the list rows
     so thats why we use same_length. */
         maplist(same_length(Rows), Rows),
     /* The concatenation of all elements of the list is an
     intenger between in range [1,9].*/
```

```
        append(Rows, Vs), Vs ins 1..9,
    /* We use all_distinct with maplist to defind that every
    row can't have the same number in above range twice. */
        maplist(all_distinct, Rows),
    /* transpose turns the rows into columns so the columns
    are now rows. */
        transpose(Rows, Columns),
    % so we use maplist with all distinct now for the columns.
        maplist(all_distinct, Columns),
    % Now we give names in each rows.
        Rows = [As,Bs,Cs,Ds,Es,Fs,Gs,Hs,Is],
    % and we split each block by three elements
        blocks(As, Bs, Cs),
        blocks(Ds, Es, Fs),
        blocks(Gs, Hs, Is).

/* We need to define the usage of blocks &    */
/* We need to pass three Empty list's []      */
/* The reason of using block is to ensure each block 3x3 to contain
distinct intengers in every combination in range [ ins 1..9]
*/
/* Example [1,2,3,4,5,6,7,8,9] == TRUE;  */
/* Example [1,2,3,3,5,6,7,8,9] == FALSE;      */
/* Example [9,1,2,4,5,6,8,7,3] == TRUE;       */
blocks([], [], []).
blocks([N1,N2,N3|Ns1], [N4,N5,N6|Ns2], [N7,N8,N9|Ns3]) :-
      all_distinct([N1,N2,N3,N4,N5,N6,N7,N8,N9]),
      blocks(Ns1, Ns2, Ns3).

/* For clear console screen */
cls :- write('\e[H\e[2J').
```

```
/* hardest
          Rows=[[_,_,_,_,2,_,7,_,8],
                [_,7,_,_,_,9,_,1,_],
                [_,3,1,_,_,_,_,4,_],
                [9,_,_,4,5,_,_,2,6],
                [_,_,_,_,_,_,_,_,_],
                [_,_,_,_,_,_,_,_,_],
                [_,_,_,_,_,_,9,6,_],
                [_,2,_,_,_,_,_,_,_],

     [_,_,_,1,7,_,_,_,_]],sudoku(Rows),maplist(label,Rows),
     maplist(portray_clause,Rows).
*/
```