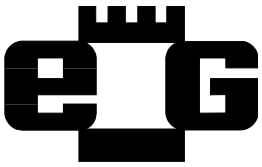

Computer News

CQL FOR DUMMIES

EMIL VLASÁK

Computer Revolution of the Endgame Study

In the last 20 years, computers have changed many human intellectual activities. This includes chess, chess composition and chess study. Aspects of the computer revolution of the chess study have – step by step – been covered in my previous columns. Let us recapitulate the most prominent topics.

(1) The new **chess engines** of GM-strength have dramatically improved the soundness of studies (EG170, EG172). However as a result also heavy non-artistic positions have been composed and published as studies (EG173).

(2) **Endgame databases** (EGTBs) changed the results of endgame theory, especially in pawnless endings. To our surprise, the RBN-RB material (no matter if same-color-bishops) and also the RNN-RB one (EG168) are generally won.

Our Codex was changed, too, because of problems with the **originality**. Since the year 2008, computer databases explicitly have no copyright in relation to chess study originals. So a database position, i.e. analyzed and evaluated by a computer, can today be published as an original study (EG173).

(3) According to Kasparov, **computer databases** are the most important development for chess since invention of typography. And in the endgame study this can be said about the **Harold van der Heijden collection** (EG174).

(4) To make this picture complete, a tool named Chess Query Language (**CQL)** has to be mentioned.

So far it has not been discussed it in my column because of its difficulty. I estimate that no more than twenty chessplayers are able to use it, worldwide. So let us try to increase this sad count a little in this column.

What is CQL?

CQL is a tool allowing searches for complicated chess themes in computer databases. Here are several examples: “stalemate with several pins”, “staircase manoeuvre” or “8th WCCT theme”. Even top commercial software such as ChessBase or ChessAssistant have, so far, not been able to master such tasks.

CQL stands for Chess Query Language. *For experts: the abbreviation CQL is a small joke, imitating SQL a computer language widely used by common database applications.*

CQL is both

– a search software, specializing in endgame studies (but can be applied to chess games and problems).

– a formal language for this purpose.

The current version of CQL is 3.01 from 2004.

Yes, you have to program

Contemporary users expect that all software intelligence is encapsulated in several dialog boxes and can be assessed by mouse clicks. Unfortunately this is not possible in CQL, because chess themes are too rich and miscellaneous.

Thus there is no other way – you have to learn programming. The authors of the CQL – Gady Costeff and Lewis Stiller – did their best to simplify as much as possible. There is also an additional tool **VisualCQL**, written by me, to help you.

A complete CQL program is called a CQL **script** (or CQL **query**).

What do you need?

CQL does not need complicated subfolders on your hard drive. Simply create the C:\CQL directory and copy all requested files there. The list of files follows, for details about all items see the Link section.

Maybe some administrator rights problems could arise in the foolish Windows Vista, but that is beyond the scope of this article.

(1) First, you need some **endgame study database in PGN format** as a reference. The best choice is Harold's database, but it is not free. The current version III (and probably also the coming version IV) is delivered in PGN, so there are no additional problems.

Older versions are in the CBF or CBH formats. See the EG*174* computer column to learn how to convert them into the PGN format.

To make things easy, rename the database "heijden.pgn".

(2) The **CQL engine** is freely downloadable from Gady's website. You will get a single archive cql.zip, containing the engine itself (cql.exe) plus a manual and examples.

Extract all these to your CQL folder.

(3) The **Visual CQL** tool is freely downloadable from my website.

Extract all the files from the archives vcql.zip (user interface) and pgnv.zip (PGN viewer) to your CQL folder again.

(4) It is a good idea to create **a shortcut** to C:\CQL\VisulCQL.exe on your desktop.

Right click an empty desktop area. A local menu appears, select New >> Shortcut and browse to find and select the appointed exe file.

Now you are ready for your first attempts.

A Quick Overview

Using the created shortcut, run the VisualCQL software.

(1) The CQL skeleton and basic rules

The VisualCQL looks like a small text editor. **A basic script skeleton** is immediately ready to use. Without the VisualCQL, you would have to write it from scratch again and again.

```
(match
:pgn heijden.pgn     ;the name of the PGN file to look for studies
:output result.pgn   ;the name of the result file, must be nonempty

  (position

  );end position
); end match
```

Remember several important rules.

(1.1) A **semicolon** starts a **comment** – the CQL engine ignores it to the end of line.

My recommendation: continuously add comments to your CQL scripts! These can be saved to your hard drive for future modifications and re-use. After several months you will not understand your own ideas or constructions without comments!

(1.2) The starting skeleton is well commented, maybe even over-commented. So you can easily understand it. "heijden.pgn" is the name of the database to be searched, while the results are saved in the "result.pgn" database. The result.pgn database will be emptied and filled with the new search results!

(1.3) The **round brackets** "(" and ")" have to be strictly in pairs and well structured. Hard to explain it exactly here, remember for example mathematical formulas. Several brackets "(" could be open, but they have to closed ")" in a reverse order. The whole text from "(" to its corresponding ")" is called a **block**.

(1.4) Every CQL script is created by the large **"match" block**, which is rather a formal matter. It forms something like a container for other more actual blocks. In the starting skeleton you can recognize a single **"position" block** inserted (one could use more than one).

"Position" blocks are very important part of CQL scripts, acting as search units.

(2) Standard menu functions in the VisualCQL

The VisualCQL upper menu is almost standard.

(2.1) The CQL scripts can be saved (File >> Save As, File >> Save) and loaded again (File >> Open). "*.Cql" is always used as the file extension.

(2.2) Familiar clipboard functions (Edit >>Cut /Copy /Paste) are ready to ease your work.

(3) Special chess functions

Complete anticipation tests can be executed from the VisualCQL software, without leaving its environment. For this reason the following three special chess functions are available:

(3.1) The **Palette** icon helps you to insert CQL functions and skeletons, without writing them from a keyboard.

(3.2) The **Runner** icon starts Gady's and Lewis' searching engine.

(3.3) After the Runner has been finished, the **Eye** icon allows you to examine searching results.

(4) Example scripts

A lot of working examples have been supplied by the authors. Try the File >> Open dialog to see them. Examples have intuitive self-explaining names.

Pieces and board areas in the CQL

It is necessary to start with some dry definitions. Pieces and board areas create the basic CQL "alphabet". They are very easy to understand.

(1) Standard Pieces

```
KQRBNP for White pieces
kqrbn for Black pieces
```

(2) Nonstandard Pieces

```
A/a  White/Black any piece
M/m  White/Black major piece
I/i  White/Black minor piece
```

```
U    any piece
.    no piece (empty square)
?    any piece or empty square
```

(3) Board Areas

Like in the usual chess notation, board areas are written after the pieces in CLQ scripts. The exact "grammar" of them is sometimes difficult. Maybe an examination of several examples is the best way to learn it.

Remember:

- the square brackets ("[", "]") enclose a list of alternatives.
- the dash ("-") is for a range, abbreviating "from-to"
- and finally the question mark ("?") is for any row or column.

```
.e4           e4 square is free
[RB]c?        White rook or bishop on c-file (c1,c2..)
pd-e4-5       Black pawn in the center (d4, e4, d5, e5)
P[a2-7,h2-7] White pawn on the a- or h- column
```

Using the Palette

You do not need to memorize nonstandard codes and keywords. Forget these, you can use the Palette.

Let us start with a small exercise.

(1) By a mouse click, locate the cursor at a free line in the "position" block.

(2) Click the upper Palette icon. You get the Palette.

(3) The Palette is a window with a table. The columns of this table are theme-structured, with fitting captions.

(4) In the second column (named "Pieces") click for example the "MinorWhite" cell.

(5) The Palette disappears and you have under the cursor the correct code "I" for a White minor piece.

The Palette incorporates the whole CQL language. So it acts as a help and the same time it saves your time – you need not write keywords and skeletons.

Your first example

It is time to create a simple, but nontrivial example.

We are searching for studies with the White king traveling from a1 to h1, or in the opposite direction. A simple example, but without CQL almost impossible or at least requiring a considerable amount of work. As we already know, the foundation stone of CQL is a "position" block, acting as a search unit. Several "position" blocks are taken by default as the "and" operator. It means all these position blocks have to match and only then the whole study is selected as suitable.

For advanced users: a more complex logical structures can be created, using logical operators in the CQL language – ":and"," :or", ":not".

The query finds studies where we have a white King at a1, but also sometimes a white King at h1:

```
(match
 :pgn heijden.pgn
 :output result.pgn
 (position Ka1)    ; WKa1
 (position Kh1)    ; WKh1
                   ; by default both positions have to match
 )                 ; end match
```

A note. I have reduced comments and formatting to make the whole CQL script more synoptic. Corresponding tip: In the Palette press the Alt key while clicking a keyword. This way a reduced one-line version of keywords or skeletons will be inserted .

Now you are ready to run the script prepared above.

(1) Click the Runner icon. A **CQL console** appears, where you can observe a searching process. It takes under one minute even on slow computers. As a result, the CQL engine reports 13 studies found in my HH III database.

(2) Close the CQL console pressing any key.

(3) Click the Eye icon. You get a **PGN viewer** allowing to examine studies, found by the engine.

The PGN viewer

After you click the Eye icon, you get the PGN viewer.

From the upper, it is created by:

(1) A game list.

(2) A chessboard on the left.

(3) A notation window on the right.

(4) Bottom buttons.

Using intuitive mouse clicks and double clicks, studies can be loaded from the list and their moves can be replayed. The notation window is click sensitive. The bottom buttons' clicks are for advanced functions, for example jumps in the game.

The "MATCH" comment

The CQL engine adds by default comments "MATCH" to fitting moves of matched studies.

Tip: There are bottom buttons to jump to the previous/next "MATCH" comment in a notation. In some cases this can be a very useful function.

Quick keyboard controls

The PGN viewer is also designed for rapid control from a numeric keyboard. This way, advanced users can complete the whole examination of matched studies with right-hand- fingers, without needing to move this hand.

Tip: Park the mouse cursor on any button on the bottom edge of the PGN viewer. Wait a while and you get a bubble help with its keyboard equivalent.

So you can:

– change studies using the 8/2 keys.

– replay moves through the 4/6 keys (plus the 5 key in branches).

– jump to the begin/end of the study through the 1/7 keys .

– jump to previous/next "MATCH" location through the 3/9 keys.

Tip: On notebooks with a reduced keyboard, the corresponding left "QEYC" keyboard block can be used as well.

Sublines and the dumb method

The CQL script just tested works only with main lines. To also search sublines, you have to add the keyword ":variations" in the position blocks. These now should look like:

```
(position
```

```
  :variations
  Ka1)
(position
  :variations
  Kh1)
```

Remember: To make things well-ordered, keywords and functions begin in the CQL with a colon.

But running such a script, you not only get intended studies with the king's a1-h1 travel. You also find studies, in which for example the Ke1 goes to a1 in one line and to h1 in another line.

Unfortunately CQL does not have good subline management. This is not a problem of CQL, but rather of PGN. Also the PGN format has no standard way to differ thematic and technical sublines.

In recent years I acted as a judge or a director of several international tourneys. New PGNs which I received from composers often use the comment "main" at a start of thematic lines. But it is not accepted as a standard, and Harold's database uses it only partly.

The dumb method

You have two ways to overcome this problem.

(1) To use more complicated CQL constructions. These will be discussed later.

(2) To use simple "superfluous" CQL scripts and examine the results "by hand".

The second way I call the **dumb method**. The dumb method helps in many cases

– where CQL doesn't work well (later I will give an example)

– if an exact query would be difficult to write.

I myself use it often.

For example, the ":variations" version of our Ka1-h1 script generates only 31 studies. It takes only several minutes to replay all them in a blitz tempo.

Other basic keywords and functions

Besides pieces and board areas, there are several other basic functions. I only give here the most important ones, for the full survey see the Palette.

```
:wtm :btm                  White to move  Black to move
:mate :stalemate :check :nocheck
                           These don't need a comment
:initial :terminal         The initial or terminal position of the study
:enpassant :noenpassant    The next move is/isn't enpassant
:movefrom :moveto :promote Examples follow
```

Examples

```
:promote [RBN]  ;the next move is any white's underpromotion
:promote [BN]a8 ;the next move is promotion to B or N on a8
:movefrom U?8   ;any piece will move from the 8th rank
:moveto .a2     ;any piece will move to the empty square a2
:moveto rh1     ;any (white) piece takes a black rook on h1
```

Remember, CQL has no deeper chess intelligence. All searched themes have to appear on the board. If a PGN line ends only one move before an obvious mate or stalemate (which is often the case), they are not recognized by the CQL engine.

Programming by example, shifts and flips

A composer is frequently interested in some pieces-pattern, no matter where exactly placed on the board. The CQL language has a very easy way to master such cases. You can describe only one

certain pattern and generalize it, using shift/flip functions. Run the Palette to observe the rich repertoire of shift/flip functions.

(1) As an example, let's find studies with the well-known stalemate idea like Ke1/Ra2 Se3.

To save space, a formal "match" block, identical to the one given above, is omitted in the next examples.

```
(position
   :stalemate
   Ke1 ne3 r?2
   :shifthorizontal   ;move the pattern horizontally
);end position
```

This easy script finds all such stalemates with the White king on the 1st rank.

Remember, shift/flip functions – although seemingly easy – are really hidden cycles. In other words, the "position" filter has to be computed many times. So the searching time increases considerably.

(2) When adding a ":flip" function, you will find all studies with the king on the edge of the board.

```
(position
   :stalemate
   Ke1 ne3 r?2
   :shifthorizontal    ;move the pattern horizontally
   :flip               ;and flip the patter in addition
);end position
```

(3) And by using the ":shift" function instead of the ":shifthorizontal" one, you get also studies with central stalemates, like Ke6-Ke4 Ra7 Se8.

You get also several nonsenses as a "bonus", because the shift operators don't work correctly near the board's edge. Use a midboard position instead (Ke2 ne4 r?3) and voila – only correct studies are found!

For advanced users: The shift/flip concept is an excellent way to get useful results quickly. But technically it cannot master more independent shifting patterns.

So the CQL language (from version 3.0 on) has also a classical cycle statement (with variables and keywords ":forany", ":tagmatch") at one's disposal. In this way cycles can be nestled. It solves also the problem how to distinguish two same pieces.

Counting in CQL scripts

You can count a number of pieces, a power of pieces, a number of attacks to several square and a number of matches. Examples follow.

```
:piececount R 2          ;there are exactly 2 White rooks
:piececount p 3 5        ;3,4 or 5 Black pawns
:piececount [Aa] 0 5     ;5 or less pieces on the board
:piececount I[c1-8] 1 2  ;1 or 2 white minor pieces on the c-file
```

A power of pieces uses the usual weights Q=9, R=5, B=N=3, P=1. Remember K=0.

```
:power a 6                 ;Black's power is 6
:power [M] 5 10            ;White has R (5), Q (9) or RR (10)
:powerdifference U 1       ;White has material advantage 1 pawn
:powerdifference [MmIi] -2 ;Black (a negative value)is an exchange ahead
:attackcount A rh3 1 2     ;White pieces attack bRh3 1 or 2 times
```

Repeated matches could be tested using a ":matchcount" function.

For example you need studies with repeated return the white king to the "a1" square.

The code for a five- or more times return (in the main line) follows.

```
(position
    :movefrom K?? ;the King moves
```

```
    :moveto ?a1   ;to a1 - no matter if empty or capture
    :matchcount 5 20)
```

For advanced users: The ":matchcount" function allows the counting of only one event. Using variables again (functions ":accumulate" and ":sumrange"), more independent events can be counted and evaluated. The situation is similar to ":forany" cycles.

Finding pins

CQL has a rich repertoire of "ray" functions. which test if the set of pieces is on one line.

Examples.

```
:ray (q A K)          ;any white piece pinned by Black's queen
:raydiagonal(Q B n k) ;the 4 pieces - in the given order - on any diagonal
:rayvertical(K . . k) ;Kings in vertical opposition
                      ;with at least two empty squares
:ray (a A K)          ;any white piece pinned ??
```

A small testing question – in the last example, must it be a pin? Of course not, because of the "a" could be for example a knight. This problem is elegantly solved by the ":rayattack" function.

```
:rayattack a A K      ;it is surely a pin
:rayattack a A K 2 3  ;2-3 pins on the board
```

Several times I have been asked to find a trendy theme "stalemate with pins". Using the ":rayattack" function, it is very easy. Here is the code for 3 and more pins.

```
(position :wtm
          :stalemate
          :rayattack (a A K) 3 5)
```

You try and experiment modifying this script.

It seems that no study with 5 pins was composed, perhaps because it would need at least one promoted piece.

Only one study with 4 pins is found (Martsvalashvili 1987), but it is a wrong result. The CQL engine doesn't find this better setting: Jasik, StrateGems 1998. For some reason, this is only recognized as a 3-pins-study. Yes, any software is full of bugs and again the dumb method helps you to overcome them.

Sequence and gapped sequence

The very useful ":sequence" function culminates our beginners' course. Shortly, it defines a continuous (uninterrupted) sequence of positions.

Its structure is a little complicated. In the basic "position" block the ":sequence" function is nested with a lot of other "position" blocks. That's why I give the full script listing. This example searches for a queen's staircase.

```
(match
:pgn heijden.pgn
:output result.pgn
 (position        ;the basic position container
  :shift          ;to find also "shifted" positions
  :sequence       ;sequence function
   (              ;start sequence bracket
     (position Qa2)
     (position)   ;Black's move, any position matches it
     (position Qb2)
     (position)
     (position Qb3)
     (position)
     (position Qc3)
     (position)
```

```
      (position Qc4)
    )             ;end sequence bracket
  )               ;end basic position bracket
)                 ;end match bracket
```

Also a ":gappedsequence" function is available in the CQL. It works alike the ":sequence", but this sequence of positions can be interrupted. In other words, move gaps are allowed.

":Gappedsequence" is also an easy way to overcome our old problem with the King's travel.

```
(position
  :variations         ;search also in sublines
  :gappedsequence
    ((position Ka1)
     (position Kh1))  ; end gappedsequence
);end position
```

The correct direction a1-h1 is found here and in addition, sublines are mastered correctly.

For advanced users

What to do next?

(1) There is a complex **":relation"** function. It allows you to search for studies with similar positions, differing only in small detail. It was developed to master the 7th WCCT theme, which cannot be probably made using normal CQL functions. The complete script was published in EG151.

(2) You should study and test functions with variables (":forany", ":accumulate"). Writing the script for the 8th WCCT, I have had to use the ":forany" cycle, because two independent pieces act here. For all that, the result was only rough and it had be to combined with the dumb method.

(3) Instead of writing very difficult scripts, an easy way to combine elements that make up a theme is to run a script with the description of the first element and use the result-file as the input for the query with the description of the second element.

And my final information for you: I love CQL, but surprisingly there are themes which cannot be found even using it.

A small example: CQL is probably unable to find studies with vertically symmetvrical position somewhere in the solution. To master such specialties, some ultimate CQL version (or another future tool) will be needed to deal with variables, math functions and low-level access to squares and pieces,. Actually, the latest CQL improvements head primarily to this direction, Unfortunately, it would mean higher difficulty for users.

Links

http://www.rbnn.com/cql/ Gady Costeff. All about CQL – download, manual, examples, articles.

http://www.vlasak.biz/vcql.htm Emil Vlasák. All about CQL. Visual CQL.

http://home.concepts.nl/~he16442/ Harold van der Heijden – database.

Costeff,G.: CQL – Chess Query Language, EG*151*. Gady's introductory article.