CQL 2

Case Query Language

Version 2.3 — 19.2.1999

TECINNO
Gesellschaft für innovative Software
Systeme und Anwendungen mbH

Sauerwiesen 2
67661 Kaiserslautern

Tel.: 06301 606 400
Fax: 06301 606 409
E-Mail: cbrworks@tecinno.com
Internet: http://www.tecinno.com/

Changes and Extensions

BNF Notation Extension

To avoid the awkward description of the common pattern x {y x }* we abbreviate that by x \ y
Example:
< list with delimiters > ::= <entry> { <delimiter> entry }*
can be written as
< list with delimiters > ::= <entry> \ <delimiter>

Latest Changes:

1.0	9.3.1998	initial revision
1.1	12.3.1998	minor changes
1.2	13.3.1998	Slot definitions have weights
1.3	2.6.1998	Date-Types added
1.4	7.7.1998	Similarities added Slot Definitions are now optional! (empty models problem)
1.5	18.7.1998	added missing measure description to the deftype-Statement changed range of <minute> and <second> to 0 .. 59
1.6	19.6.1998	added descriptions of return codes
1.7	29.6.1998	corrected syntax description for notation above
1.8	23.8.1998	Relevant added <query> and <get all case references> now distinct between including and not including subclasses <parameter string> added
1.9	23.10.1998	number of cases relevant for computing entropy may be given on query
2.0	21.11.1998	completely revised CQL to CQL2
2.0.1	11.12.1998	enhanced Error-Messages
2.0.2	18.12.1998	extended Query for using rules
2.0.3	11.1.1999	extended descriptive model for values
2.0.4	12.1.1999	added keyword ruled for slot values slot values changed
2.0.5	16.1.1999	missed out notations added corrected some errors in notations
2.1	21.1.1999	new commands for requesting parts of a domain model added placement and ranking
2.2	28.1.1999	added description for rules added filters
2.2.1	16.1.1999	properties will be send with basic model
2.3	17.1.1999	added basic user management for client access
2.3.1	19.1.1999	corrected type definition for print names and annotations

Assumptions:

- Class definitions, type definitions, and objects have to be defined, before they are referenced using identifiers.

Syntax

Domain Model

<descriptive model> ::=
    <basic model>
    {<measure definition>}*
    {<rule definition>}*

<basic model> ::=
    [<model properties>]
    {<type definition>}*
    {<value definition>}*
    {<slot definition>}*
    {<class definition>}+

<model properties> ::=
    <properties>
    version <string>
    serial <string> “.”

Case Data

<case data> ::= {<add case>}+

Class Definitions

<class definition> ::=
    defclass <class identifier> a_kind_of {<class denoter>}
    [<class properties>]
    [is_case_class]
    [slots {<slot identifier>}+]
    [placement { precede | inplace | succeed }]
    [<comment>]
    {<print name>}*
    {<annotation>}* "."

<class identifier> ::= <symbol>

<class denoter> ::= class | <class identifier>

<class properties> ::= <properties>

Slot Definitions

<slot definition> ::=
    defslot <slot identifier> of <class identifier>
    {type <type identifier> | class <class denoter>}
    [weight <weight>]
    [mandatory]
    [ranking <positive integer>]
    [not_discriminant]
    {<print name>}*
    {<question>}*
    {<annotation>}* "."

<slot path> ::= <slot identifier> \ ">"

<slot identifier> ::= <symbol>

<question> ::= question <language identifier> <string>

Type Definitions

<type definition> ::=
deftype <type identifier> a_kind_of <type identifier>
[<type properties>]
[range <range restriction>]
[currentMeasure <measure identifier>]
{<print name>}*
{<annotation>}* "."

<type identifier> ::= <symbol>

<type properties> ::=
<properties>
[unit <string>]

Range Restriction

<range restriction> ::= <interval range definition> | <enumeration definition> | <taxonomy definition>

<interval range definition> ::= "[" {<basic value>} ".." {<basic value>} "]"

<enumeration definition> ::= "(" {<basic value>}+ ")"

<taxonomy definition> ::= "[" <taxonomy tree> "]"

<taxonomy tree> ::= <basic value> ["[" {<taxonomy tree>}+ "]"]

Value Definitions

<value definition> ::=
defvalue <value identifier> of_type <type identifier>
{<print_name>}* “.”

<value identifier> ::= <symbol>

Measure Definitions

<measure definition> ::=
defmeasure <measure identifier> { of_class <class identifier> | of_type <type identifier> }
[<measure properties>]
mode <symbol>
[parameters <parameter string>] “.”

<measure identifier> ::= <symbol>

<measure properties> ::= <properties>

<parameter string> ::= <string>

Rule Definitions

<rule definition> ::=
defrule <rule identifier> of <model identifier>
priority <integer>
[inactive] (completion_rule | adaptation_rule)
{<condition definition>}*
{<action definition>}* “.”

<model identifier> ::= <type identifier> | <class identifier>

Conditions

<condition definition> ::=
condition (<function definition> | <predicate definition> | <type convert definition>)

<predicate definition> ::=
predicate <argument definition> <predicate> <argument definition>

<function definition> ::=
function <function identifier>
result <variable identifier>
[<argument definition>]

<type convert definition> ::=
typeconvert variable <variable identifier> type <model identifier>

<predicate> ::= <comparator> | is_regular | is_special | is_kind_of

<function identifier> ::= “““ (add | div | mult | sub) “““

Actions

<action definition> ::=
action
(<assignment definition> | <inconsistency definition> | <typechange definition> |
<similaritychange definition> | <weight definition> | <filterchange definition>)

<assignment definition> ::=
assignment <slotaccess> “=” <argument definition>

<inconsistency definition> ::=
inconsistency “(“ {<slotaccess>}* “)”

<typechange definition> ::=
type <slotaccess> kind_of (<variable>| <concept>)

<similaritychange definition> ::=
similarity <similarity identifier> type <type identifier>

<weight definition> ::=
weight model <model identifier>
slot <slot identifier>
value <positive real>

<filterchange definition> ::=
filter <slotaccess> <filter>

Arguments

<argument definition> ::=
argument (<slotaccess> | <constant> | <variable> | none)

<variable> ::= variable <variable identifier>

<constant> ::= constant <type identifier> <basic value>

<concept> ::= concept <concept identifier>

<slotaccess> ::= <adaptationslotaccess> | <completionslotaccess>

<adaptationslotaccess> ::= adaptationSlot (query | target | retrieved) <slot path>

<completionslotaccess> ::= completionSlot <slot path>

Objects & Values

<object> ::=
    <class identifier> <object identifier>
    [<slot instantiation> \ ","]
    {<printname>}*
    {<annotation>}*

<slot instantiation> ::= <slot identifier> ":" [ruled] <slot value>

<slot value> ::=
    <basic value> | <relational value> | <set> | <interval> | unknown | undefined | relevant | notApplicable

<relational value> ::= <object identifier>

<set> ::= "{" (<basic value> | <relational value>) \ "," "}"

<interval> ::= “[“ <basic value> .. <basic value> “]”

<basic value> ::=
    <integer> | <real> | <string> | <symbol> | <boolean> | <date> | <time> | <timestamp>

<object identifier> ::= <symbol>

Cases

<case> ::=
    defcase [<case number>]
    [<case mode>]
    [<case properties>]
    objects {<object> \ “;” } "."

<case reference> ::=
    case_reference <case number>
    [<case mode>]
    {<printname>}*

<case mode> ::= confirmed | unconfirmed | obsolete | protected

<case properties> ::= <properties>

Comments

<free comment> ::= <comment> “.”

Command Definitions

The server responds to a client call as soon as it has recognized the ESCAPE Character (Character 27). Results always are followed by the ESCAPE Character (Character 27).

Calls

<get descriptive model> ::= get_descriptive_model "."

<get basic model> ::= get_basic_model "."

<get types> ::= get_types "."

<get classes> ::= get_classes "."

<get values> ::= get_values "."

<get slots> ::= get_slots "."

<query> ::=
send_query <no. of cases to retrieve> <threshold> [complete_query]
[case_class <class_identifier> [including_subclasses]]
[case_references_only | adapt_cases]
[questions <no. of paths to return> [<no. of relevant cases for question computation>]]
objects { <query object> \ ";" } "."

<get all case references> ::=
get_all_case_references [of_class <class_identifier> [including_subclasses]] "."

<get case> ::=
get_case { <case number> | first } "."

<add case> ::=
add_case <case> “.”

<update case> ::=
update_case <case> “.”

<delete case> ::=
delete_case <case number> "."

Results

<get descriptive model result> ::=
returns <return code> descriptive_model <descriptive model>
end

<get basic model result> ::=
returns <return code> basic_model <basic model>
end

<get types result> ::=
returns <return code> types {<type definition>}*
end

<get classes result> ::=
returns <return code> classes {<class definition>}+
end

<get values result> ::=
returns <return code> values {<value definition>}*
end

<get slots result> ::=
returns <return code> slots {<slot definition>}*
end

<query result> ::=
returns <return code> [<error description>]
[questions {<ranked slotpath> \ ","}]
[cases { <query result case> }*]
[query <case>]
end

<get all case references result> ::=
returns <return code> [<error description>] [case_references { <case reference> }*]
end

<get case result> ::=
returns <return code> [<error description>]
[case <case> next_case_no <case number>]
end

<add case result> ::=
returns <return code> [<error description>] case_ID <case number>
end

<update case result> ::=
returns <return code> [<error description>]
end

<delete case result> ::=
returns <return code> [<error description>]
end

Miscellaneous Definitions

<return code> ::= <integer>
<error description> ::= <string>

User Management Commands

The server responds to a client call as soon as it has recognized the ESCAPE Character (Character 27).
Results always are followed by the ESCAPE Character (Character 27).

Calls

<get user list> ::= get_user_list "."

<add user> ::= add_user <user name> <password> <access level> “.”

<delete user> ::= delete_user <user name> “.”

Results

<get user list result> ::=
    returns <return code>
    user_list
    { <user name> <password> <access level> \ , }
    end

<add user result> ::=
    returns <return code>
    end

<delete user result> ::=
    returns <return code>
    end

Elements

<user name> ::= <string>

<password> ::= <string>

<access level> ::= <integer>

Supportive Definitions

<properties> ::=
creator <string>
<creation date>

<creation date> ::= <date>

<print name> ::= print_name <language identifier> <string>

<annotation> ::= annotation <language identifier> <string>

<query object> ::= <class identifier> <local object identifier> [<query slot> \ ","]

<query slot> ::= <slot identifier> ["(" [<weight>] [<filter>] ")"] ":" [ruled] <slot value>

<query result case> ::= <similarity value> ":" { <case> | <case reference> }

<filter> ::= <comparator> | <interval> | <set>

<comparator> ::= equal | not_equal | less | less_or_equal | greater | greater_or_equal

<language identifier> ::= <symbol>

<comment> ::= comment <string>

<ranked slotpath> ::= <positive real> ":" <slot path>

<weight> ::= <positive real>

<case number> ::= < positive integer>

<no. of cases to retrieve> ::= < positive integer>

<no. of paths to return> ::= < positive integer>

<no. of relevant cases for question computation> ::= < positive integer>

<similarity value> ::= {0.0 .. 1.0}

<threshold> ::= {0.0 .. 1.0 }

General Definitions

<date> ::= date ”(“ <year> <month> <day> “)”
<time> ::= time “(“ <hour> <minute> <second> “)”
<timestamp> ::= timestamp ”(“ <year> <month> <day> <hour> <minute> <second> “)”

<year> ::= < positive integer >
<month> ::= {1 .. 12}
<day> ::= {1 .. 31 }
<hour> ::= {0 .. 23}
<minute> ::= {0 .. 59}
<second> ::= {0 .. 59}

<integer> ::= {["-"] <positive integer>} | "0"

<positive integer> ::= {"1".."9"} {"0".."9"}*

<real> ::= ["-"] <positive real>

<positive real> ::= {"0".."9"}+ ["." {"0".."9"}+] ["E" ["+"|"-"] { "0".."9"}+]

<boolean> := true | false

<symbol> ::= <string>

<string> ::= """ { <string character> | """" }* """

<string character> ::= any ASCII character except """

Return Codes

Values returned by CQL-Server as part of the result of a CQL-Command.
negative return values are errors, positive return values are warnings

0 Request successful

Errors

-1 Syntax error
-2 unknown class identifier
-3 unknown type identifier
-4 unknown slot identifier
-5 invalid case class
-6 ambiguous identifier
-7 inappropriate value
-8 case not found

Warnings

1 class identifier changed to case class
2 found mandatory slots being undefined. case set unconfirmed
3 slots referencing unavailable cases set to unknown
4 removed unavailable references from filter

Notation-Index