Temporal Query Language

version 2.2

TI = time interval (for example [0 – 2, 1 – 3, 1 – 5, 5 – 10])
CTI = computed time interval (for example [0 – 10])
TP = time point (internally a time interval with same time of start and end)
CPT = examples: 12345 or 1234F
ICD9 = examples: 022, 022.2, 022.22, E123, E123.1, E123.12, V01, V01.1, V01.01
NOTE_TP = TP with payload information [TP, NOTE_ID]
ICD9_TI = TI with payload information [TI, ICD9_ID]
MAX = Integer.MAXIMUM (maximum positive number)
MIN = Integer.MINIMUM (minimum negative number)

Automatic Result type conversions

TP => TI[TP, TP]
TP => BOOLEAN[is TP empty]
TI => BOOLEAN[is TI empty]
TI => TP (undefined)
TI => CTI
BOOLEAN => TIMELINE
BOOLEAN => TP (undefined)
NOTE_TP => TP
ICD9_TI => TP

Skipped patients

A query can skip patients. Reasons for skipping a patient are:
- FAIL PATIENT command
- querying for a feature that is flagged as incomplete (if at least one event for a feature does not have a valid time (dates before birth, dates after death, etc.), whole feature will be flagged as incomplete and querying for it will skip the patient
- querying negation commands (NOT, INVERT, NOTE(NOT(, BEFORE(X, Y)-()) if the whole feature set is missing in patient's records. For example querying for RX=1192 will skip patient if patient has no RX codes.

Comments

Comments are portions of the query that are not evaluated.

1. Line comment is a comment starting with // and ending with end of line
2. Multi line comment is a comment starting with /* and ending with */
3. Line starting with a # character denotes a comment

Examples:

```
var a = ICD9=250.00 // this is a comment
// also a comment
/* multi line comment
another multi line comment */
#comment
$a
```

Variables

Variable definition:

var variable=UNION(ICD9=200.2, ICD9=100.1)

Variable usage:
INTERSECT($variable1, $variable2)

Saving variables to server

var domain_name.variable_name=UNION(ICD9=200.2, ICD9=100.1);
Domain name is an identifier string that allows grouping variables into cohesive units.

Variables that contain other variables will be stored after evaluating the nested variables, so that there are no dependencies between stored variables.
Example:
var variable1=UNION(ICD9=100.1, ICD9=200.2)
var variable2=INTERSECT(ICD9=200.3, ICD9=300.3)
var john.variable3=UNION($variable1, $variable2)

john.variable3 will be stored as UNION(UNION(ICD9=100.1, ICD9=200.2), INTERSECT(ICD9=200.3, ICD9=300.3))

Accessing saved variables

$my_workspace.variable_name

When declaring a variable and not using it in the same query, an error is thrown (no query), but the variable will be persisted.

Example:
var john.variable = ICD9=200.5
This will define a global variable, but will not store it. Variable will be stored only after you call the variable with .finalize, .store or .save suffix. Already defined variables cannot be overwritten. Once the finalize was called, the variable is read only. This also means that when you attempt to redefine a variable that was already saved, this will throw an error.

Example:
var john.variable = ICD9=200.5
$john.variable.finalize

Defining variable without storing it:
var john.variable = ICD9=200.5
$john.variable

Stored variables cannot be redefined.

Basic temporal operations

TIMELINE

Returns a single TI containing patient's whole timeline.
Example: TIMELINE
Returns TI

ATC

Returns TI during which the patient had at least one RXNorm code belonging to the specified ATC
Example: ATC=”XXX”
Returns TI

ENCOUNTERS

A day during the patient had at least one encounter (only a visit counts as an encounter).
Returns a list of TI that are computed in such a way that each TI is exactly 1 day long and there is no overlap between TI.

NOTES

Returns all time instances when the patient had text notes.

PATIENTS(X, Y, Z)

Returns patients with the specified patient IDs
Example: AND(PATIENTS(1, 3, 5, 7, 12), ICD9=250.00)
Returns BOOLEAN

SNOMED=

Returns a list of TI during which the patient had the specified SNOMED code
Example: SNOMED=12345
Returns TI

SNOMED

Returns a list of all SNOMED TI for a patient
Example: SNOMED
Returns TI

DEPARTMENT=

Returns a list of all TI for a patient during which he was in the specified department
Example: DEPARTMENT=”xxxxx”
Returns TI

CPT=

Returns a list of TI during which the patient had the specified CPT code
Example: CPT=12345
Returns TI

CPT

Returns a list of all TI for all CPT codes of a patient. Is equivalent to UNION(CPT=…..)
Example: CPT
Returns TI

VISIT TYPE=”xxxx”

Returns time interval during which the patient had the specified visit type.
Returns TI

ICD9=

Returns a list of ICD9_TI during which the patient had the specified ICD9 code
Example: ICD9=E123.22
Returns TI

ICD9

Returns a list of all ICD9_TI for a patient. Equivalent to UNION(ICD9=….)
Example: ICD9
Returns TI

ICD10=

Returns a list of ICD10_TI during which the patient had the specified ICD10 code
Example: ICD10=E123.22
Returns TI

ICD10

Returns a list of all ICD10_TI for a patient. Is equivalent to UNION(ICD10=….)
Example: ICD10
Returns TI

NOTE TYPE=

Returns a time interval during which the patient had note with the specified note type.

Example: NOTE TYPE="Progress Note, Outpatient"

NOTE command allows querying text commands within a note with the specified note type.
Example: NOTE(NOTE TYPE="Progress Note, Outpatient", TEXT="diabetes")

Returns NOTE_TP

PRIMARY

Only usable on a single ICD9 or ICD10 code. Returns TI for which the patient had the ICD9 or ICD10 code mentioned as primary diagnosis
Example: PRIMARY(ICD9=E123.22)

To return times when the code was mentioned but was not a primary code:
INTERSECT(ICD9=E123.22, INVERT(PRIMARY(ICD9=123.22)))
Returns TI

RX=

Returns a list of RXNORM_TI during which the patient had the specified RXNORM code
Example: RX=122
Returns RXNORM_TI

RX

Returns a list of all RXNORM_TI for a patient. Equivalent to UNION(RX=...)
Example: RX
Returns RXNORM_TI

DRUG

Allows querying additional properties of a drug (drug status and drug route)
Example: DRUG(RX=122, STATUS="discontinued", ROUTE="intravenous")
DRUG(RX=122, STATUS="discontinued")

DRUG(RX=122, ROUTE=”intravenous”)
Returns TI

TEXT=”   “

Returns a list of NOTE_TP where the patient had specified text mentioned without negation or family history
Returns NOTE_TP

!TEXT=”   “

Negated text was present in notes.
Returns NOTE_TP

~TEXT=”   “

Family history text was present in notes
Returns NOTE_TP

Text field limitations

Whenever using the text fields (for example: TEXT=”something”, ATC=”something”), the text field between the quotation marks can contain any character, except for quotation mark character and a dollar sign. These are reserved for the internal use in the language.

NOTE(TEXT=”   “, TEXT=”   “)

All the text specified has to occur in one note.
NOTE also accepts boolean operators (AND, NOT, OR) as their parameters.
NOTE itself behaves in a same way as the command NOTE(AND(TEXT=” “, TEXT=” “))
NOTE(OR(TEXT=” “, TEXT=” “)) returns all patients that have any of the specified texts. This is identical to OR(TEXT=” “, TEXT=” “)
NOTE(NOT(AND(TEXT=” “, TEXT=” “))) is identical to NOT(NOTE(TEXT=” “, TEXT=” “))

Boolean notes allow more complex queries such as:
NOTE(AND(TEXT=”diabetes”, OR(TEXT=”treated”, “treatment”, “controlled), NOT(TEXT=”uncontrolled”))

Returns NOTE_TP

YEAR

Returns a TI during which the patient had any code during the specified year
Example: YEAR=2008
Caution: Year only contains the intervals explicitly defined in visits and notes. There is no computation of missing intervals, so using EXTEND BY could lead to an interval that has an undefined YEAR.
Returns TI

GENDER

Example: GENDER=”MALE”
Return BOOLEAN

RACE

Example: RACE=”WHITE”
Return BOOLEAN

ETHNICITY

Example: ETHNICITY=”LATINO”
Return BOOLEAN

DEATH / DEAD

Returns the TP of patient's death
Example: DEATH
Returns TP

NULL

Returns an empty TI

VITALS

Returns the time points at which the vitals had specified values
Example: VITALS(“Pulse”, 60, 100)
VITALS(“Pulse”)
If no parameters are used, all times when the patient had vitals taken are returned
Example: VITALS
Returns TP

LABS

Returns the time points at which the labs had specified values.
Example: LABS(“WBC”, “NORMAL”)
LABS(“WBC”)
LABS(“WBC”, 2.5, MAX)
If no parameters are used, all times when the patient had labwork done are returned
Example: LABS
Returns TP

AGE

Returns the time interval during which the patient was of a specified age range.
Example: AGE(30 years, 35 years)

The difference between INTERVAL(30 years, 35 years) and AGE(30 years, 35 years) is, that AGE will return only the intervals during which the patient had some event recorded, whereas INTERVAL will construct the intervals regardless of whether there were some events or not.

Returns TI

Boolean operations

Boolean operations operate on other boolean operations or basic temporal operations. If boolean operations operate on a mixture of basic temporal operations and boolean operations, appropriate temporal operation is performed first, then evaluation of the boolean operations is performed.

AND

Performs a boolean AND operation on other boolean operations or returns TRUE if the input is at least one TI

Example: AND(CPT=1234,CPT=222,OR(CPT=2345,CPT=435))
Results:
CPT=1234 = 1-5
CPT=222 = 3-7
CPT=2345 = 2-3
CPT=435 =5-7
result = AND(1-5,3-7,OR(2-3,5-7))
result = AND(true,true,OR(true,true))
result = AND(true,true)
result = true
Returns BOOLEAN

OR

Performs a boolean OR operation on other boolean operations or true if the input is at least one interval

Example:OR(CPT=222,CPT=333,AND(CPT=444,CPT=555))
Results:
CPT=222 = 1-5
CPT=333 = 2-8
CPT=444 = 2-4
CPT=555 = 6-9
result = OR(1-5,2-8,AND(2-4,6-9))
result = OR(1-8,AND(true,true))
result = OR(1-8,false)
result = OR(true,false)
result = true
Returns BOOLEAN

NOT

Performs a boolean NOT operation on another single boolean operation or returns true if the input is an empty TI

Patient's record: CPT 3 = 1-5
CPT 4 = 10-12
CPT 1 = 7-11
Example:NOT(CPT=1)
result = false

Example: NOT(AND(CPT=4,CPT=1))
result = NOT(AND(true,true))
result = NOT(true)
result = false
Returns BOOLEAN

IDENTICAL

Tests two commands whether they are completely identical (same number of intervals having the same starts and ends)

The difference between EQUAL and IDENTICAL is that EQUAL returns any intervals of same starts and ends, but IDENTICAL returns all the intervals provided there are the same numbers of intervals with the same starts and ends for both arguments.

Example: IDENTICAL(INTERVAL(100, 200), UNION(INTERVAL(100, 200), INTERVAL(250, 300)))
result = false

EQUAL(INTERVAL(100, 200), UNION(INTERVAL(100, 200), INTERVAL(250, 300)))
result = [100, 200]

TI operations

TI operations operate on TI, if the input is a boolean operation, true is converted to the TI of the whole patient's timeline and false is converted to an empty interval.

START

Prerequisites: Performs TI => CTI
Returns a set of TI with the time coordinates [START, START]

Patient's record: CPT 3 = 1-5, 2-7, 9-10
Example:START(CPT=3)
result = [1-1, 2-2, 9-9]
Returns TP

END

Prerequisites: Performs TI => CTI
Returns a set of TI with the time coordinates [END, END]

Patient's record: CPT 3 = 1-5, 2-7, 9-10
Example:END(CPT=3)
result = [5-5, 7-7, 10-10]
Returns TP

COUNT

Does not perform computation of intervals before evaluation. Computation is performed on the result if both COMMAND_EVALUATE and COMMAND_INTERVAL are specified as intervals to return.
Returns a TI for which the number of specified commands was in the specified range.
Command accepts following configurations of parameters:

    COUNT(COMMAND_EVALUATE, MIN, MAX)
    COUNT(COMMAND_EVALUATE*, COMMAND_INTERVAL*, TYPE , MIN, MAX)

Asterisk denotes which interval to return in the second case (command_interval or command_evaluate). Default is to return command_evaluate.

COMMAND_EVALUATE = Condition in which count we are interested in
MIN / MAX = interval of min / max counts. MIN and MAX values are accepted
COMMAND_INTERVAL = section of patient's timeline for which we want to evaluate the COUNT command
TYPE = SINGLE (a single interval in patient's timeline must have the corresponding counts with double counting) / ALL (cumulative counts for all intervals with no double counting)

Example: COUNT(OR(~TEXT=”diabetes”, ~TEXT=”cancer”))*, INTERSECT(ICD9=250), ALL, 5, MAX)
Takes TI for which the patient had ICD9=250 and counts the occurrences of “diabetes” or “cancer” in family history and returns all the ICD9=250 intervals if cumulatively there were at least 5 counts.

INTERVAL(X, Y)

Given two time points, constructs a TI. If the parameters are TI, all combinations of START of TI1, END of TI2 will be performed for which it is true, that START of TI1 <= END of TI2.
Interval parameters can be numeric, in which case it constructs an interval from the time points specified.

Interval command is also compatible with time commands.

Example: INTERVAL(3 years, 4 years)
result = [(3*365*24*60), (4*365*24*60)]

Patient's record: CPT 3 = 1-5
CPT 4 = 10-12
Example: INTERVAL(START_OF(CPT=4), END_OF_RECORD)
result = INTERVAL(4-4, 12-12)
result = 4 – 12
Returns TI

INTERVAL(X, Y, PAIRS)

Evaluates X and Y as series of pairs.
Result = UNION(INTERVAL(START(X1), START(Y1)), INTERVAL(START(X2), START(Y2)), INTERVAL(START(X3), START(Y3))) for all X1 <= Y1 and X2 <= Y2 and X3 <= Y3 etc.

Without the PAIRS parameter, the following evaluation is made:
Result = UNION(INTERVAL(START(X1), START(Y1)), INTERVAL(START(X1), START(Y2)), INTERVAL(START(X1), START(Y3)), INTERVAL(START(X2), START(Y1)), INTERVAL(START(X2), START(Y2)), INTERVAL(START(X2), START(Y3)), etc.)

Returns TI

INTERSECT

Prerequisites: Performs TI => CTI
Returns an intersection of multiple TI. If there is no intersection, returns an empty TI.

Patient's record: CPT 3 = 1-5
CPT 4 = 10-12
CPT 1 = 7-11
Example:INTERSECT(CPT=1,CPT=4)
result = INTERSECT(7-11,10-12)
result = 10-11
Returns CTI

UNION

Prerequisites: Performs TI => CTI
Performs an union of multiple TI

Patient's record: CPT 3 = 1-5
CPT 4 = 10-12
CPT 1 = 7-11
Example:UNION(CPT=3,CPT=4)
result = UNION(1-5,10-12)
result = 1-5,10-12
Returns CTI

FIRST MENTION

Returns a first time interval for a given expression

Patient's record: CPT 3 = 1-5, 10-12
CPT 4 = 10-12
CPT 1 = 7-11
Example:FIRST_MENTION(CPT=3)
result = 1-5
Returns Single TI with min start and min end

First mention can work also in a particular context returning first mention of a term in a particular interval.
Patient’s record: CPT 2 = 5-10, 20-30
CPT 3 = 1-3, 15-22
Example: FIRST MENTION(CPT=3, CPT=2) (find first mention of CPT3 in CPT 2)
result = 20-22
Returns first mention (if any found of CPT=3 within CPT=2 time interval)

LAST MENTION

Returns the last time interval for a given expression

Patient's record: CPT 3 = 1-5, 10-12
CPT 4 = 10-12
CPT 1 = 7-11
Example:LAST_MENTION(CPT=3)
result = 10-12
Returns Single TI with max start and max end

Last mention can work also in a particular context returning last mention of a term in a particular interval.
Patient’s record: CPT 2 = 5-10, 20-30
CPT 3 = 2-8, 31-35
Example: LAST MENTION(CPT=3, CPT=2) (find first mention of CPT3 in CPT 2)
result = 5-8
Returns first mention (if any found of CPT=3 within CPT=2 time interval)

EXTEND BY (RESIZE)

Prerequisites: Performs TI => CTI
Extend the start and end of each TI by a specified amount of time units.

EXTEND BY(TI, time_start, time_end)

EXTEND BY(TI, 0, START + 10)

EXTEND BY (TI, END – 10, 0)

If the intervals overlap after the extension, they will be merged. Intervals that would extend patient's time-line will be truncated so that they are contained in the time line.

Extends start of the interval by time_start (negative towards past, positive towards future) and end of the interval by time_end.

Patient's record: CPT 3 = 1-5, 10-12

Example:EXTEND_BY(CPT=3,-2, 2)
result = (1–2)-(5+2),(10-2)-(12+2)

result = (-1)-(7),(8)-(14)
result = 0-7, 8-12
Returns TI

EVALUATE / EVAL

Parameters: EVALUATE(TYPE, COMMAND)

TYPE:

INTEGER	evaluates the first N patients and returns how many were true
TIME	returns as many patients as possible in the specified time
CACHED	evaluates cached patients only

LIMIT

Parameters: LIMIT(TYPE, COMMAND)

TYPE:

INTEGER	returns the specified number of patients LIMIT(1000, ICD9=200.00) => returns first 1000 pids
TIME	returns as many patients as possible in the specified time
CACHED	evaluates cached patients only

ESTIMATE

Parameters: ESTIMATE(TYPE, COMMAND)
Only returns the positively evaluated PIDs. Estimates the total cohort statistics based on evaluated pids.

TYPE:

INTEGER	returns the specified number of patients ESTIMATE(1000, ICD9=200.00) => returns first 1000 pids
TIME	returns as many patients as possible in the specified time
CACHED	evaluates cached patients only

EQUAL

Parameters: EQUAL(TI1, TI2)

Returns only the intervals that have the same starts and ends.

BEFORE STRUCTURE

Before command can be invoked in the following structure:
(X) BEFORE Y*
    returns every Y which exists at least one minute after the start of X
(X*) BEFORE Y
    returns every X which exists at least 1 minute before the start of Y
(X AND Y AND Z) BEFORE A*
    returns every A that has at least 1 minute before its start at least 1 minute of X, Y and Z
(X* AND Y AND Z*) BEFORE A
    returns every X, Z for which it’s true that it exists before a start of A
(X) AND NO (Y) BEFORE A*
    returns A that has X and no Y before its start

(X*) AND NO (Y) BEFORE A
    returns X that exists before A and where there is no Y before A
(X AND Y) AND NO (Z AND B) 3 MONTHS* BEFORE A
    returns 3 months before A for which it is true that there was a X and Y but no Z and B

Note that is is not possible to return any data points from the negative mentions NO (…). Otherwise it is possible to tag any elements of the expression including the time element.

AFTER STRUCTURE

After command can be invoked in the same structure as before with the same syntax, except that the keyword AFTER is used.

BEFORE (SEQUENCE)

Performs TI=>CTI on TI1 and TI2
Returns TI (does not perform CTI on results, so if results are [2, 5] [4, 8] it will NOT compute it into [2, 8])
Parameters: BEFORE(TI1, TI2)
Asterisk denotes which parameter to return is evaluation is successful.
Returns TI1 before TI2. Can be followed by multiple parameters.
TI1 and TI2 can be followed by asterisk which designates whether to return TI1 and/or TI2 as a result of a successful query.
When no asterisk is specified and no return type is specified in other parameters, command cannot be executed.
Parameters: BEFORE(TI1, TI2)+>(-100, -1)-*<>(END+1, END+100)
+ sign before a parameter means that the TI1 has to occur in that range. If the TI1 does not happen, then the TI2 for which we evaluate will fail
- sign before a parameter means that the TI1 cannot occur in that range. If IT1 occurs, the TI2 fails

To return the range of the condition (-100, -1), asterisk needs to be added before the range:
BEFORE(TI1, TI2)+*(-100, -1)
This command returns the range of (-100, -1) before the TI2, if TI1 was located in that range

If parameters start with < TI1 had to begin in the specified range, if > TI1 had to end in that range, if <> same TI1 had to both start and end in the interval (contained by the interval).

The time range parameter +(100, 200) is always evaluated based on the TI2, so BEFORE(TI1, TI2)+(100, 200) will take the TI2, extend the START of TI2 by 100 towards the future, extend the END by 200 into the future and evaluates whether there is TI1 in that range.

BEFORE(X, X) is not a valid command since TI compression takes place in BEFORE command

Example 1

X1 = [5, 5]

Y1 = [10, 20]

Y2 = [35, 40]

BEFORE(X, Y)*-(-6, -1) = [29, 39]

As we can see parameter *-(-6, -1) is false for Y1 (X1 is within the specified range even though it should not be), but is true for Y2.

BEFORE(X, Y*)-(-6, -1) = [35, 40]

Y2 is returned since it is evaluated to true even though Y1 was evaluated to false.

Example 2

If we want to return only the intervals for which it is true that they were ALWAYS in a certain range before Y

(there is no instance of X that was not in the range), we should use:
A = BEFORE(X, Y)+*(R1, R2)
B = BEFORE(X, Y)-*(R1, R2)
INTERSECT(NOT($B), $A)
If there is at least one interval X that is not in range R1, R2, nothing will be returned.
Y1 = [10, 20]
Y2 = [30, 40]
X1 = [25, 27]
A = BEFORE(X, Y)+*(MIN, -1) = [25, 27]
B = BEFORE(X, Y)-*(MIN, -1) = [0, 9]
INTERSECT(NOT($B), $A) = []

DURATION

Performs computation of intervals first.
If COMMAND_EVALUATE and COMMAND_INTERVAL are followed by an asterisk, computation of intervals is performed on the result.

Returns a TI for which the duration of specified commands was in the specified range.
Command accepts following configurations of parameters:

    DURATION(COMMAND_EVALUATE, TYPE , MIN, MAX)
    DURATION(COMMAND_EVALUATE*, COMMAND_INTERVAL*, TYPE , MIN, MAX)

Asterisk denotes which interval to return in the second case (command_interval or command_evaluate). Default is to return command_evaluate.

COMMAND_EVALUATE = Condition in which duration we are interested in
MIN / MAX = interval of min / max counts. MIN and MAX values are accepted
COMMAND_INTERVAL = section of patient's timeline for which we want to evaluate the DURATION command. Each time interval in the section is evaluated separately and then next time interval is iterated and evaluated. ALL operator in this case will return if the patient had at desired durations cumulatively WITHIN the single time interval specified by the command.

TYPE = SINGLE (a single interval in patient's timeline must have the corresponding durations – double counting is permissible) / ALL (cumulative durations for all intervals – double counting is not permissible)

INVERT

Prerequisites: Performs TI => CTI
Inverts a TI or TP by using patient's time line as a reference. Inversion of an empty interval is patient's complete interval.

Patient's record: CPT 3 = [1-3], [7-10]
CPT 2 = []
CPT 4 = [5 – 8]
Example: INVERT(CPT=2)
result = 1-10

Example: INVERT(INTERVAL(START_OF_RECORD, FIRST_MENTION(CPT=4))
This is equivalent to no history of
result = INVERT(1-5)
result = 6-10
Returns CTI

Macro expressions

Macro expressions are sets of expression encapsulating common expression for convenience sake

RECORD START

START(TIMELINE)

RECORD END

END(TIMELINE)

HISTORY OF(X)

Patient had a history of X. Returns TI from the first mention of X until the end of record. If patient never had X, empty interval will be returned.
INTERVAL(START(FIRST MENTION(X)), RECORD END)

NO HISTORY OF(X)

Patient did not have a history of X. This will return a TI of the whole patient's time line if the patient never had X, or an interval from the beginning of the patient's time line to the development of X.
UNION(NOT(X), INVERT(HISTORY_OF(X)))

CONTAINS(X, Y)

Returns Y only if it's fully contained in X.
BEFORE(X, Y)+*<>(START, END)
CONTAINS(X, X) is invalid as there is compression in BEFORE command
Returns TI of X that which are fully contained in Y

RETURN X INTERSECTING Y

Returns full interval X if it intersects Y
BEFORE(Y, X*)+(START, END)

RETURN X INTERSECTING ANY (A, B, C)

Returns full interval X if it intersects at any point either A, B, or C.
BEFORE(UNION(A, B, C), X*)+(START, END)

RETURN X INTERSECTING ALL (A, B, C)

Returns full interval X if it intersects at any point A, B, and C. The intervals A, B, C do not have to intersect each other, but they have to intersect X
1 = BEFORE(A, X*)+(START, END)
2 = BEFORE(B, X*)+(START, END)
3 = BEFORE(C, X*)+(START, END)
4 = INTERSECT(X, 1, 2, 3)

RETURN X ALWAYS INTERSECTING Y

Returns full interval X if in patient’s timeline all X intervals intersect Y
1 = BEFORE(Y, X*)-(START, END)
2 = NOT(1)
3 = INTERSECT(X, 2)

RETURN X ALWAYS INTERSECTING ANY (A, B, C)

Returns full interval X only if every X interval in patient’s timeline intersects at some point A, B or C
1 = BEFORE(UNION(A, B, C), X*)-(START, END)
2 = NOT (1)
3 = INTERSECT(2, X)

RETURN X ALWAYS INTERSECTING ALL (A, B, C)

Returns full interval X if every X in patient’s timeline always intersects A, B and C
1 = BEFORE(A, X*)-(START, END)
2 = BEFORE(B, X*)-(START, END)
3 = BEFORE(C, X*)-(START, END)
4 = INTERSECT(1, 2, 3)
5 = NOT (4)
6 = INTERSECT(5, X)

RETURN X NOT INTERSECTING Y

Returns full interval X if it does not intersect Y
BEFORE(Y, X*)-(START, END)

RETURN X NOT INTERSECTING ANY (A, B, C)

Returns full interval X if it does not intersect A, B or C. Intervals intersecting either A, B, or C will be removed
BEFORE(UNION(A, B, C), X*)-(START, END)

RETURN X NOT INTERSECTING ALL (A, B, C)

Returns full interval X if it does not intersect Y
1 = BEFORE(A, X*)+(START, END)
2 = BEFORE(B, X*)+(START, END)
3 = BEFORE(C, X*)+(START, END)
4 = INTERSECT(1, 2, 3)
5 = INVERT(4)
6 = INTERSECT(X, 5)

RETURN X NEVER INTERSECTING Y

Returns full interval X if it does not intersect Y
1 = BEFORE(Y, X*)+(START, END)
2 = NOT (1)
3 = INTERSECT(X, 2)

RETURN X NEVER INTERSECTING ANY (A, B, C)

Returns full interval X if it never intersected in patient’s timeline A, B or C
1 = BEFORE(A, X*)+(START, END)
2 = BEFORE(B, X*)+(START, END)
3 = BEFORE(C, X*)+(START, END)
4 = UNION(1, 2, 3)
5 = NOT(4)
6 = INTERSECT(5, X)

RETURN X NEVER INTERSECTING ALL (A, B, C)

Returns full interval X if it is true that in patient’s timeline it never intersected all of A, B, and C
1 = BEFORE(A, X*)+(START, END)

2 = BEFORE(B, X*)+(START, END)
3 = BEFORE(C, X*)+(START, END)
4 = INTERSECT(1, 2, 3)
5 = NOT(4)
6 = INTERSECT(X, 5)

NEVER HAD(X)

Patient never had a condition X. Returns patient's whole time line (true) if he never had X or an empty interval / false if he had.
NOT(HAS(X))

EVENT FLOW (X)

Exports the current cohort into an EventFlow / CoCo file formats.

OUTPUT (X)

When generating the list of PIDs in the export, the PID information will also contain time intervals for when the evaluated query was true in format PID [TAB] START_TIME [TAB] END_TIME

CSV

Generates a csv file with specified list of columns.
CSV(INTERSECT(ICD9=250.00, CPT=24560), “METFORMIN”=AND(RX=1235), “DEATH TIME”=DEATH)
Apart from labeled columns, following commands will output all codes of the specified type:
ICD9, ICD10, LABS, VITALS, CPT, RX

If the column result would specify multiple time points / time intervals, they will be delimited with a semicolon
Example:
CSV(ICD9=250.00, “SURGERY”=CPT=22456)

CSV command returns all of patient’s data for the entire timeline. It is possible to specify a subset of timeline using the TIME or START / END modifiers.

Example:
CSV(ICD9=250.50, TIME=VISIT TYPE=”INPATIENT”, CPT)
For all patients that had ICD9 code 250.50 output all CPT codes they had while being inpatient.

CSV(ICD9=250.50, START=EXTEND BY(DEATH, START-30 DAYS, START), END=DEATH, ICD9, LABS)
For all patients that had ICD9 code 250.50 output all ICD9 and LABS they had 30 days before death

Cohort operations

SAME

Returns list of patients that are in both compared cohorts.
VAR cohort1= INTERSECT(ICD9=250.0, NOTES)
VAR cohort2= AND(GENDER=”MALE”, RACE=”WHITE”)
SAME($cohort1, $cohort2)

DIFF

Returns list of patients that are represented in first parameter but not in second
DIFF($cohort1, $cohort2)

MERGE

Returns list of patients added from both cohorts
MERGE($cohort1, $cohort2)

Examples:
History of disease A
    HISTORY OF(A)
No history of disease A
    NO HISTORY OF(A)
Patient with disease A taking drug B after procedure C
    BEFORE(START(INTERSECT(A, B)), START(C))
Patient with disease A never had disease B
    INTERSECT(NEVER HAD(B), HAS(A))
Patient with a history of disease B taking medication C
    INTERSECT(HISTORY OF(B), C)
Patient with no history of disease A having disease B
    INTERSECT(NO HISTORY OF(A), B)
Patient who had A maximum 6 months before developing B
    BEFORE(A, B)+*(-6*30*24*60, -1)
Patient who had A at least 6 months before developing B
    BEFORE(A, B)-(6*30*24*60, -1)+*(MIN, -6*30*24*60-1)
Patient who had A exactly 6 months before developing B
    BEFORE(A, B)+*(-6*30*24*60, -6*30*24*60)
Patient hospitalized (H) for more than 6 months following procedure A (hospitalization had to occur 24 hours after procedure A)
    DURATION(BEFORE(H, BEFORE(A, H)+*(-24*60, -1))+*<(START, END), 6*30*24*60, MAX)
Define diabetes as at least 2 mentions of ICD9=250.*, then extend it from the first mention to the end of life
    INTERVAL(FIRST MENTION(COUNT(ICD9=250.*, 2, MAX, ALL)), RECORD END)
Patient that had at least 2 mentions of family history of diabetes, take first mention of cancer and extend until end of recorded
    INTERVAL(FIRST_MENTION(INTERSECT(TEXT=”cancer”, AND(COUNT(~TEXT=”diabetes”, 2, MAX, ALL)))), RECORD_END)
Patients who took medication X within 3 months before they died
    INTERSECT(X, EXTEND BY(DEATH, -3*30*24*60, 0))
Patients who underwent surgery A and they did not take medication X for at least 1 month prior to surgery
    BEFORE(X, A*)-(-30*24*60, -1)
Patients who started taking medication X 3 months before procedure A and stopped 1 month before the procedure and they never took the medication since. We need at least 6 months of continuous follow-up data afterwards
    COUNT(ENCOUNTERS, EXTEND BY(BEFORE(X, BEFORE(X, A*)+<(-3*30*24*60, -3*30*24*60)*)->(-30*24*60, -1), 6*30, MAX, ALL)
Patients that had diabetes(D) and during diabetes were not taking metformin(M).
    INTERSECT(D, INVERT(M))
Patients who did not take medication X prior to event Y
    BEFORE(X, Y*)-(MIN, 0)
Patients who had surgery X while not being on a medication Y (X not intersecting Y)
    INTERSECT(Y, INVERT(INTERSECT(X, Y))
Patients that had lab A, B and C within 24 hours. Let's assume that labs are a single time points without duration
    INTERSECT(EXTEND(A, 0, START + 24 HOURS), EXTEND(B, 0, START + 24 HOURS), EXTEND(C, 0, START + 24 HOURS))
Patients that had at least 10 labs A within 30 days and a lab B and C done at the same time-line
    INTERSECT(B, C, COUNT(A, EXTEND(A, 0, START + 30 DAYS), 10, MAX))
24 hours during which patients had A, B, C
    EXTEND(INTERSECT(EXTEND(C, 0, START + 1 DAY), EXTEND(B, 0, START + 1 DAY), EXTEND(A, 0, START + 1 DAY)), 0, 24 HOURS)

Iterative evaluation

To evaluate each time interval in a query separately, iterative evaluation can be used. Examples where iterative evaluation allows operations that cannot be done in regular approach.

```
FOR EACH (COMMAND) AS (LABEL_MAIN) {
}
```

Takes each time interval from the command COMMAND and makes it accessible in a for loop as LABEL_MAIN

Example:

```
FOR EACH (INTERSECT(ICD9=250.50, CPT=25000)) AS (DIABETES) {
   RETURN DIABETES AS VARIABLE_1;
}
INTERSECT(VARIABLE_1, GENDER=”MALE”)
```

Takes each interval from the intersection of both codes, makes it available in a for loop under the name DIABETES. There is no algorithm in place so it just returns all the results as VARIABLE_1. VARIABLE_1 is then accessible in the global context and can be evaluated.

Commands available in the loop:

```
LABEL = COMMAND;
CONTINUE;
EXIT;
FAIL PATIENT;
CLEAR GLOBAL_LABEL;
RETURN LABEL AS GLOBAL_LABEL;
IF EMPTY (COMMAND) { ... }
IF !EMPTY (COMMAND) { … }
IF (COMMAND1) == (COMMAND2) { ... }
IF (COMMAND1) != (COMMAND2) { ... }
IF (COMMAND1) IN (COMMAND2) { ... }
IF (COMMAND1) !IN (COMMAND2) { ... }
```

LABEL=COMMAND;

Stores the result of the command into a variable accessible by the label name in the current for each loop context. Variable is not accessible in a nested for each context and is not accessible in global context. This command can be used to reassign previously assigned variable as well as global variable.

At the end of each loop, all local variables are deleted. The persistence of local variables is only within a single loop cycle.

To store value persistently use the RETURN command.

Example:

```
FOR EACH (ICD9=250.50) AS (DIABETES) {
   LONGER_THAN_3_YEARS = DURATION(DIABETES, SINGLE, 3 YEARS, MAX);
   RETURN LONGER_THAN_3_YEARS AS RESULT_1;
}
RESULT_1
```

CONTINUE;

Skips the rest of the FOR EACH loop.

Example:

```
FOR EACH (ICD9=250.50) AS (DIABETES) {
  LONGER_THAN_3_YEARS = DURATION(DIABETES, SINGLE, 3 YEARS, MAX);
  // skips intervals shorter than 3 years
  IF EMPTY(LONGER_THAN_3_YEARS) {
    CONTINUE;
  }
  // return command is skipped for shorter ones
  RETURN DIABETES AS RESULT_1;
}
RESULT_1
```

EXIT;

Stops evaluation of the FOR EACH loop and exits. All the results stored in the global variable before the exit will be returned.

Example:

```
FOR EACH (ICD9=250.50) AS (DIABETES) {
  LONGER_THAN_3_YEARS = DURATION(DIABETES, SINGLE, 3 YEARS, MAX);
  // returns only those instances that are longer than 3 years and followed by the first shorter instance
  IF EMPTY(LONGER_THAN_3_YEARS) {
    EXIT;
  }
  RETURN DIABETES AS RESULT_1;
}
RESULT_1
```

FAIL PATIENT;

Stops evaluation of the FOR EACH loop and causes the patient’s evaluation to be FALSE (empty interval).

Example:

```
FOR EACH (ICD9=250.50) AS (DIABETES) {
  LONGER_THAN_3_YEARS = DURATION(DIABETES, SINGLE, 3 YEARS, MAX);
  // returns only those patients which had all their durations longer than 3 years
  IF EMPTY(LONGER_THAN_3_YEARS) {
    FAIL PATIENT;
  }
  RETURN DIABETES AS RESULT_1;
}
RESULT_1
```

Returns only the patients which did not have any ICD9=250.50 codes shorter than 3 years.

RETURN LABEL AS GLOBAL_LABEL;

Takes a variable or a command and stores the resulting value into the global context (if FOR EACH is nested in another FOR EACH command, returned nested variable will be stored in the parent context.

Example:

```
FOR EACH (ICD9=250.50) AS (DIABETES) {
  A = INTERSECT(CPT=250000, DIABETES);
  FOR EACH (A) AS (NESTED_A) {
    B = DURATION(NESTED_A, SINGLE, 3 YEARS, MAX);
    IF NOT EMPTY(B) {
      RETURN B AS GLOBAL_B;
      }
   }
   // GLOBAL_B is accessible within this context since it was returned by the nested FOR EACH loop
   RETURN GLOBAL_B AS RESULT_1;
}
// RESULT_1 is available in global context since it was returned by the parent FOR EACH
RESULT_1
```

CLEAR GLOBAL_LABEL;

Clears the contents of the global variable.
Example:

```
FOR EACH (ICD9=250.50) AS (DIABETES) {
   A = INTERSECT(CPT=250000, DIABETES);
   FOR EACH (A) AS (NESTED_A) {
      B = DURATION(NESTED_A, SINGLE, 3 YEARS, MAX);
      IF NOT EMPTY(B) {
         RETURN B AS GLOBAL_B;
      }
      IF EMPTY(B) {
         CLEAR GLOBAL_B;
         EXIT;
      }
   }
}
// GLOBAL_B is incrementally receiving values from each iteration unless B is empty which clears all the
// previous results and exits the loop
GLOBAL_B
```

IF EMPTY (COMMAND) { … }

Executes commands in curly braces if the command evaluates to an empty interval.

IF !EMPTY (COMMAND) { … }

Executes commands in curly braces if the command evaluates to an non empty interval.

IF (COMMAND1) == (COMMAND2) { … }

Executes commands in curly braces if the command1 time intervals equal command2 time intervals

IF (COMMAND1) != (COMMAND2) { … }

Executes commands in curly braces if the command1 time intervals are not equal to command2 time intervals

Using multiple iterative loops

Iterative loops are evaluated sequentially.

In this example, first the ICD9=222.22 is pushed to a global variable DEF, then global variable DEF is used in the second loop and returned as DEF.

```
FOR EACH(ICD9=222.22) AS (ABC) {RETURN ABC AS DEF;}
FOR EACH(DEF) AS (DEF) {RETURN DEF AS DEF;}
DEF
```

This for loop will throw an error, since it relies on a variable DEF that has is unknown
FOR EACH(DEF) AS (DEF) {RETURN DEF AS DEF;}
FOR EACH(ICD9=222.22) AS (ABC) {RETURN ABC AS DEF;}
DEF

Each FOR EACH loop is evaluated, even if it does not return any results, so just the presence of a FOR EACH loop modifies the query.
FOR EACH loop is ignored if it does not contain FAIL PATIENT OR does not RETURN a variable used elsewhere.