**MXQL (Version 0.8)**

MXQL heißt Medical XML Query Language und ist eine Suchsprache die von verschiedenen Query-Engine-Implementationen des PaDaWaN-Frameworks verarbeitet können werden soll.

MXQL modelliert die folgenden Aspekte:

* welche Patienten werden abgefragt:
	+ die Patientenselektierung erfolgt über die Definition von notwendigen Attributen.
	+ boolesche Operatoren können die Selektionskriterien verknüpfen
	+ es kann verschiedene weitere Kontext-Bedingungen geben, wie z.B. absolute Beschränkungen der Attribute Selektionskriterien auf bestimmte Wertebereiche, relative Abhängigkeiten der Attribute untereinander oder Bedingungenen dass mehrere selektierte Attribute zum selben Fall oder Dokument gehören müssen
* welche Daten der selektierten Patienten werden zurückgegeben
	+ alle Daten (unter Berücksichtigung der jeweiligen Einschränkungen der Attribute) der Selektionskriterien werden zurückgegeben
	+ weitere optionale Attribute
	+ sollen von den angegebenen Attributen weitere Daten wie der Messzeitpunkt oder die zugehörige DocID ausgegeben werden
* wie werden die Daten zurückgegeben
	+ Formatierung bei Rückgabe von mehreren Instanzen eines Attributes. Sollen die verschiedenen Instanzen in dieselbe Zelle exportiert werden oder sollen pro Instanz mehrere Rückgbaezeilen erzeugt werden.
	+ Sortierung

Das Ergebnis einer Query ist eine Tabelle, bzw. ein Objekt vom Typ „Result“ in dem die Ergebnisse der Query repräsentiert sind, mit Spaltenheadern und Zelleninhalten. Die Inhalte der Zellen des Results sind Objekte vom jeweils passenden Typ des in der Zelle enthaltenden semantischen Typs, d.h. Texte sind Strings, Zahlen sind Integers, Doubles und Longs und Datumsangaben sind Dates.

Eine MXQL-Query ist als XML-String repräsentiert. Jedes XML-Element der Query stellt ein QueryStructureElement dar.

Ein Beispiel einer MXQL-Query in dem Patienten abgefragt werden die die Diagnose Herzinsuffizienz leiden und die entweder im Arztbrief erwähnt haben oder als ICD10-Diagnose kodiert haben ist folgende:

<Query>

 <Or>

 <Attribute project=“Arztbriefe“ extID=“Brieftext“ contains=”Herzinsuffizienz” />

 <Attribute project=“Diagnose“ extID=“I50“/>

</Or>

</Query>

Die verschiedenen QueryStructureElement-Typen bilden eine Klassenhierarchie, so dass mögliche Eigenschaften vererbt sind (d.h. z.B. können alle Elemente potenziell das „active“-Attribut haben).

Alle Timestamp-Angaben in den folgenen Elementen können eine der folgenden Formatierungen besitzen: „yyyy-MM-dd hh:mm:ss“, „yyyy-MM-dd hh:mm“, „yyyy-MM-dd“, „dd.MM.yyyy hh:mm:ss“, „dd.MM.yyyy hh:mm“, „dd.MM.yyyy“.

**<StructureElem>**

Ein StructureElem ist eine abstrakte Klasse und taucht in konkreten MXQL-Queries niemals auf.

Parameter:

active:

Datentyp: Boolean Default: TRUE

Bedeutung: Bei einer Belegung mit FALSE wird dieses Element und alle Kinderelemente ignoriert, so als wären sie gar nicht vorhanden.

comment:

Datentyp: String

Bedeutung: Ein beliebiger String der zu Dokumentationszwecken verwendet werden kann.

optional:

Datentyp: Boolean Default: FALSE

Bedeutung: Bei einer Belegung mit TRUE muss dieses QueryElement bei der Selektion von Suchtreffern nicht in Betracht gezogen werden. Das Element dient nur zur Konfiguration der Rückgabespalten der Suche. Ohne dieses Element werden also immer genausoviele Treffer gefunden wie mit ihm. Da auch QueryStructureContaineElems diesen Member erben, können auch ganze Suchäste optional gemacht werden. Bei einer Belegung mit TRUE werden Einschränkungen die im Subast der Query eventuell gefordert werden wie z.B. eventuelle contentOperatoren weiterhin ausgeführt, d.h. die optionalen Rückgabezellen sind nur gefüllt, wenn die Bedingungen des Subasts erfüllt sind.

SubElemente:

**TempOpAbs:**

Ein absoluter temporaler Operator schränkt das Attribut (bzw. alle in diesem QueryStructureElem enthaltenen Attribute) auf ein absolutes Zeitinterval ein. Diese Möglichkeit der Einschränkung ist alternativ zu einer Suche bei der ein Date-Attribut (z.B. der Aufnahme-Zeitpunkt) als explizites Attribut in die Suche mit integriert wird. Ein Attribut kann auch mehrere absolute Contraints haben. Dann werden die verschiedenen absoluten Constraints verodert.

Parameter:

minDate: Datentyp: Timestamp.

Das eingeschränkte Attribut muss einen Messzeitpunkt nach diesem Datum besitzen.

maxDate: Datentyp: Timestamp.

Das eingeschränkte Attribut muss einen Messzeitpunkt vor diesem Datum besitzen.

**TempOpRel:**

Ein relative temporaler Operator schränkt das Attribut (bzw. alle in diesem QueryStructureElem enthaltenen Attribute) auf Zeiten ein relativ zu einem anderen Attribut.

 Parameter:

 refElem: Datentyp: String

 Ein Alias eines anderen Attributs. Das Attribut mit dem alias muss in der Query existieren und darf nicht optional sein, sonst ist die Query nicht valide. Es dürfen keine Attribute aus Subqueries referenziert werden.

 dayShiftMin, monthShiftMin, yearShiftMin, dayShiftMax, monthShiftMax, yearShiftMax: Datentyp: Integer, Default: 0

 before: Datentyp: Boolean Default: FALSE

 Entspricht einer Konfiguration mit dayShiftMin: -INTMAX. Dieses Flag ist nicht kompatibel mit anderen Shift-Angaben

 after: Datentyp: Boolean Default: FALSE

Entspricht einer Konfiguration mit dayShiftMax: INTMAX. Dieses Flag ist nicht kompatibel mit anderen Shift-Angaben

Das eingeschränkte Attribut muss in einem Interval liegen dass durch den Messzeitpunkt des referenzierten Attributs determiniert wird. Das Interval hat als Ankerpunkt den Messzeitpunkt des referenzierten Attributs und wird durch die Min-Shifts nach hinten (in die Vergangenheit) und durch die Max-Shits nach vorne (in die Zukunft) aufgespannt. Die jeweiligen Shifts können auch negativ sein. Z.B. wird ein relatives Interval von mindestens 10 Tagen vor dem referenzierten Attribut und maximal 10 Tage danach wird durch „dayShiftMin: -10, dayShiftMax: 10“ ausgedrückt. Ein Attribut kann auch mehrere relative temporale Contraints haben. Dann werden die verschiedenen relativen Contraints verodert.

**<StructureContainingElem>**

StructureContainingElem können andere Objekte beinhalten, so dass die gesamte Query einen Baum darstellt. Diese Klasse ist ebenfalls abstrakt und wird nie instaziiert. Sie erbt von StructureElem.

**<Query>**

Das Root-Objekt einer Query. Die Root erbt von QueryAnd, d.h. alle Elemente die sich direkt in der Query befinden werden beereits VerANDed, d.h. man braucht kein weiteres explizites QueryAnd.

Parameter:

pid:

Datentyp: Boolean Default: FALSE

Bedeutung: Soll eine zusätzliche Spalte mit der PID ausgegeben werden?

onlyCount:

Datentyp: Boolean Default: FALSE

Bedeutung: Soll statt der Ergebniszeilen mit den konkreten Daten nur eine Zeile mit nur einer Spalte mit dem Header „Count“ und der einzigen Zelle in der die Anzahl an gefundenen Objekten steht ausgegeben werden?

limitResult:

Datentyp: Integer Default: 0

Bedeutung: Die Ergebnismenge wird maximal auf die angegebene Menge an Ergebniszeilen beschränkt. Beim Wert 0 wird dieser Parameter ignoriert und alle Ergebnisse zurückgegeben.

distinct:

Datentyp: Boolean Default: FALSE

Bedeutung: Soll die Ergebnis-Menge distincted werden auf PIDs? Es werden von mehreren möglichen Fakten eines Patienten diejenigen zurückgegeben mit dem kleinsten Value. Für Strings ist dies ein entsprechend äquivalenter Operator zu <.

version:

 Datentyp: String Default: 0.8

Bedeutung: Die Versionsnummer des Modells der vorliegenden Query. Alle neuen Queries werden mit der aktuellen Version versehen.

**<Attribute>**

Attributes erben von StructureElem. Es sind die Hauptelemente von Queries die die eigenltich zu suchenden Elemente beschreiben.

Parameter:

extID:

Datentyp: String

Bedeutung: Mit der ExtID und der Domain wird das Attribut im Katalog identifiziert. Diese beiden Elemente müssen immer angegeben werden.

domain:

Datentyp: String

Bedeutung: Mit der ExtID und der Domain wird das Attribut im Katalog identifiziert. Diese beiden Elemente müssen immer angegeben werden.

elementID:

Datentyp: String

Bedeutung: Über die ElementID kann ein Element von anderen QueryElementen referenziert werden (z.B. für relative temporale Relationen).

desiredContent:

Datentyp: String

Bedeutung: Der DesiredContent gibt an womit die Value eines Info-Eintrages belegt sein muss damit ein Ergebnis für das Gesamtergebnis erzeugt werden kann. Durch den „contentOperator“ wird genauer spezifiziert wie der desiredContent verarbeitet werden muss.

Bei Katalogeinträgen vom Typ „Text“ können im „desiredContent“ weitere Operatoren kodiert sein:

* Ein einzelnen Wort bedeutet dass dieses Wort als ganzes Token (umgeben von Leerzeichen) im durchsuchten Text vorkommen muss
* Mehrere Wörter bedeutet dass alle diese Wörter ungeachtet ihrer Position und Reihenfolge im durchsuchten Tex vorkommen müssen
* Das Zeichen „\*“ bedeutet einen Wildacrd Operator. Anstelle des „\*“ können beliebig viele Zeichen stehen. Ansonsten verhält sich das gesuchte Wort wie die Suche nach einem einzelnen Wort
* Umklammerung mit „[]“ bedeutet dass die enthaltenen Tokens im Text in einer nahen Umgebung stehen müssen (maximal 7 Tokens voneinander entfernt)
* Umklammerung mit „[]“ mit anführender Zahl ( „[n Wort1 Wort2]“) bedeutet dass die enthaltenen Tokens maximal n Tokens weit entfernt im durchsuchten Text vorkommen müssen.
* Umklammerung mit „[]“ mit anführenden + („[+ Wort1 Wort2]“) bedeutet dass die Tokens in exakt dieser Reihenfolge im Text vorkommen müssen
* „[+ Wort1 Wort2]“ und „[n Wort1 Wort2]“ lassen sich auch zu „[n+ Wort1 Wort2]“ kombinieren.

Bei Katalogeeinträgen vom Typ „DateTime“ muss die Belegung des desiredContnt bei den entsprechenden contentOperatoren folgendermassen kodiert sein:

* EQUALS | LESS | LESS\_OR\_EQUAL | MORE | MORE\_OR\_EQUAL: “yyyy-MM-dd HH:mm:ss”
* BETWEEN: “yyyy-MM-dd HH:mm:ss…yyyy-MM-dd HH:mm:ss”
* PER\_YEAR | PER\_MONTH | PER\_INTERVAL: TODO

Bei Katalogeinträgen vom Typ “Number” muss die Belegung eine Zahl sein, wobei Kommazahlen wahlweise per Punkt oder per Komma getrennt sein dürfen. Intervalle für den Operator BETWEEN werden mit drei Punkten („…“) getrennt.

contentOperator:

Datentyp: EQUALS | LESS | LESS\_OR\_EQUAL | MORE | MORE\_OR\_EQUAL | BETWEEN | CONTAINS| CONTAINS\_NOT | CONTAINS\_POSITIVE | CONTAINS\_NOT\_POSITIVE | PER\_YEAR | PER\_MONTH | PER\_INTERVAL | NOT\_EXISTS | EXISTS

Default: EXISTS

Bedeutung: Durch den ContentOperator wird spezifiziert wie der „desiredConent“ verarbeitet werden soll.

EQUALS: alle Ergebnisse müssen gleich dem String (der Zahl) im „desiredContent“ sein.

LESS: alle Ergebnisse müssen kleiner als die Zahl im „desiredContent“ sein.

LESS\_OR\_EQUAL: alle Ergebnisse müssen kleiner oder gleich der Zahl im „desiredContent“ sein.

MORE: alle Ergebnisse müssen grösser als die Zahl im „desiredContent“ sein.

MORE\_OR\_EQUAL: alle Ergebnisse müssen grösser oder gleich der Zahl im „desiredContent“ sein.

BETWEEN: alle Ergebnisse müssen zwischen dem „desiredContent“ und dem „desiredContentBetweenUpperBound“. Die Grenzen werden wie bei MORE und LESS interpretiert.

CONTAINS: alle Ergebnisse müssen den String im „desiredContent“ enthalten.

CONTAINS\_NOT: Das Attribut muss vorhanden sein, alle Tokens aus derm „desiredContent“ dürfen jedoch nicht im Value enthalten sein.

CONTAINS\_POSITIVE: alle Ergebnisse müssen den String im „desiredContent“ enthalten. Die Vorkommnisse des Strings dürfen im zu durchsuchenden Text jedoch nicht verneint sind. Diese Option benötigt eine etwas ausgefeiltere linguistische Verarbeitung um Negationen zu entdecken.

CONTAINS\_NOT\_POSITIVE: Kombination von CONTAINS\_NOT und CONTAINS\_POSITIVE

PER\_YEAR: alle Ergebnisse sind distinct auf das Jahr. Diese Option ist alternativ zum „IDFilter“ vom Typ „year“. Im „desiredContent“ kann mit zwei Jahreszahlen ein Interval angegeben werden auf welches Jahreinterval die Verteilung angewendet werden soll.

PER\_MONTH: alle Ergebnisse sind distinct auf den Monat. Das Jahr nach dem Eingeschränkt werden soll ist im „desiredContent“ angegeben.

PER\_INTERVAL: dieser Operator darf nur verwendet werden, wenn das QueryAttribut in einer statistischen Query eingebettet ist.

NOT\_EXISTS: der String im „desiredContent“ darf nicht im zu durchsuchenden Text vorkommen.

EXISTS: es wird keine Einschränkung vorgenommen. Der „desiredContent“ wird ignoriert. Das Attribut muss einfach nur vorhanden sein damit hiermit Ergebnisse gefunden werden können.

reductionOp:

Datentyp: Max | Min | Earliest | Latest | None

Default: None

Bedeutung: ähnlich einem „distinct“-Member von anderen Query-Elementen. Der ReductionOp gibt an ob die Ergebniss-Menge reduziert werden soll. Mit „None“ werden alle Ergebnisse als Zeilen zurückgegeben.

Bei „Max“, „Min“, „Earliest“, „Latest“ wird nur der jeweilige der Werte zurückgegeben.

multipleRows:

Datentyp: Boolean Default: FALSE

Bedeutung: Wenn man als „reductionOp“ „None“ angegeben hat und man für ein Attribut mehrere Ergebnisse erhält wird hiermit angegeben ob die weiteren Ergebnisse jeweils in verschiedene Zeilen mit jeweils einem Ergebnis geschrieben werden oder ob alle Ergebnisse mit „,“ getrennt in dieselbe Zelle einer Zeile geschrieben werden soll. Mehrere Attribute mit „multipleRows“ auf TRUE zu setzen ist nicht erlaubt. Wenn für das Attribut „infoDate“, „caseID“ oder „docID“ angewählt ist, dann werden auch diese Werte entsprechend mit aufgeteilt bzw. mit „,“ getrennt in eine Zelle geschrieben.

valueInFile:

Datentyp: Boolean Default: FALSE

Bedeutung: Soll der Ergebnistext direkt in der Ergebnistabelle stehen soll oder ob der Text in ein File geschrieben werden soll und in der Ergebnistabelle ein Link auf dieses File erscheint.

infoDate:

Datentyp: Boolean Default: FALSE

Bedeutung: Soll im Ergebnis eine zusätzliche Spalte erzeugt werden soll in der das Messdatum der Information steht.

caseID:

Datentyp: Boolean Default: FALSE

Bedeutung: Soll im Ergebnis eine zusätzliche Spalte erzeugt werden soll in der die CaseID der Information steht.

docID:

Datentyp: Boolean Default: FALSE

Bedeutung: Soll im Ergebnis eine zusätzliche Spalte erzeugt werden soll in der die DocID der Information steht.

displayValue:

Datentyp: Boolean Default: TRUE

Bedeutung: Soll im Ergebnis die Spalte mit dem Wert der Information überhaupt angezeigt werden oder nicht.

onlyDisplayExistence:

Datentyp: Boolean Default: FALSE

Bedeutung: Soll im Ergebnis in der Spalte mit dem Wert der Information der wirkliche Werte der Information oder nur ein „x“ angezeigt werden. Dies ist praktisch bei grossen Textattributen

searchSubEntries:

Datentyp: Boolean Default: wenn es sich beim Katalogeintrag um einen Bool-Typ handelt: TRUE ansonsten FALSE

Bedeutung: Soll bei der Suche nur nach Informationen des angegebenen KatalogEintrages gesucht werden oder auch nach allen Kinderknoten des KatalogEintrags.

elementID:

Datentyp: String Default: NULL

Bedeutung: Diese ID wird bei andere QueryElems eingetragen, wenn ein QueryAttribute referenziert werden soll (z.B. bei relativen Vergleichen wie QueryValueComp). Die elementID ist unique in einer Query. elementIDs von Macros werden nicht betrachtet.

minCount:

Datentyp: Int Default: 0

Bedeutung: Das Attribut muss mindestens so häufig vorhanden sein.

maxCount:

Datentyp: Int Default: Int.Max

Bedeutung: Das Attribut darf höchstens so häufig vorhanden sein.

countType:

Datentyp: PID | CaseID | DocID | Year Default: PID

Bedeutung: Die Member minCount und maxCount beziehen sich auf die Häufigkeit beim jeweiligen Patienten/Fall/Dokument/Jahr

extractionMode:

 Datentyp: NextNumber | BetweenHits | None Default: None

 Bedeutung: Nur bei Textsuchen wird bei:

NextNumber: Die zurückgegebene Value ist nicht der gesamte Text mit eventuellem Highlighting, sondern die nächste Zahl nach dem Hit.

BetweenHits: Die zurückgegebene Value ist nicht der gesamte Text mit eventuellem Highlighting, sondern der Text zwischen zwei Hits.

SubElemente:

**HeaderFormula:**

**ListSelector:**

**ValueCompare:**

Ein relativer Operator schränkt das Attribut auf Werte ein relativ zu einem anderen Attribut.

 Parameter:

 refElem: Datentyp: String

 Ein Alias eines anderen Attributs. Das Attribut mit dem alias muss in der Query existieren und darf nicht optional sein, sonst ist die Query nicht valide. Es dürfen keine Attribute aus Subqueries referenziert werden. Das referenzierte Attribut muss denselben Katalogeintrag darstellen wie das referenzierende Attribut (Es können nur Werte vom selben Typ verglichen werden).

Der Vergleich mit dem anderen Attribut wird anhand des ContentOperators ausgeführt. In diesem sind in diesem Modus nur noch die Operatoren MORE, LESS, EQUAL, MORE\_OR\_EQUAL, LESS\_OR\_EQUAL erlaubt.

**<And>**

Ands erben von StructureContainingElem. Alle Kinderelemente die nicht „optional“ sind müssen erfüllt sein, damit eine Ergebniszeile zurückgegeben wird.

Parameter:

powerSet:

Datentyp: Boolean Default: FALSE

Bedeutung: Dieser Member ist nur für die Verteilungssuche relevant. Er gibt an ob die Kinderelemente zu einer Potenzmenge verarbeitet werden um die Ergebnistabelle zu erzeugen.

**<Or>**

Or, siehe And, aber ohne den Member “powerSet”.

**<IDFilter>**

IDFilter erben von QueryStructureContainingElem. Die Filter können auf den Typ „PID“, „CaseID“, „DocID“ oder „Year“ gesetzt werden. Sie erzwingen eine Gruppierung der Ergebnisse einer Zeile. Mit einem Filter auf „CaseID“ müssen Ergebnisse die Eigenschaft erfüllen, dass alle Kinder des Filters (QueryAttributes, etc.) im selben Fall gefunden werden müssen. „DocID“ erzwingt analog dass die Ergebnisse im selben Dokument enthalten sein müssen. „Year“ erzwingt analog dass die Messzeiten der enthaltenen Kinder im selben Jahr gewesen sein müssen.

Parameter:

distinct:

Datentyp: Boolean Default: FALSE

Bedeutung: Soll die Ergebnis-Menge distincted werden auf den ID-Typ des jeweiligen Filters (PID, CaseID, DocID)? Es werden von mehreren möglichen Fakten eines Falles/Dokumentes diejenigen zurückgegeben mit dem kleinsten Value. Für Strings ist dies ein entsprechend äquivalenter Operator zu <.

filterIDType:

Datentyp: PID | CaseID | DocID | Year

Default: PID

Bedeutung: siehe obige Beschreibung. Ein IDFilter mit PID ohne vorgegebene IDs ist eigentlich sinnlos, da eine QueryRoot bereits auf die gemeinsame PID joined.

SubElemente:

**ID:**

Die ID-Subelemente dienen dazu den Filter weiter auf bestimmte IDs (CaseIDs, PIDs, DocIDs) oder auch Zeiten einschränken zu können. Mehrere ID-Subelemente mit derselben value-ID und verschiedenn times werden verodert. Die verschiedenen value-IDs wirken auch als Oder.

Parameter:

value:

Datentyp: Number

Bedeutung: eine ID (PID, CaseID, DocID) die als erlaubtes Ergebnis für Ergebnisse gelten darf. Der Filter erzeugt nur Zeilen mit einer der gegebenen IDs. Dieser Member muss bei einer ID vorliegen.

time:

Datentyp: Timestamp

Bedeutung: Ein Zeitstempel der angibt dass Ergebnisse (Messzeitpunkt) zur entsprechenden ID auch nur zu diesem Zeitpunkt gefunden werden dürfen. Dieser Member kann bei einer ID vorliegen muss jedoch nicht. Der Zeitstempel kann ein Datum sein, in diesem Fall müssen die Messzeitpunkte gefundenen Daten an diesem Tag liegen, oder der Zeitstempel kann eine sekundengenaue Datums-und-Zeit-Angabe sein. Die Angabe von Tagesgenauigkeit oder Sekundengenauigkeit muss für alle Zeitstempel des Filters gleich sein. Entweder haben alel IDs des Filters eine Time-Einschränkung oder keine. Durch die Angabe von Time-Einschränkungen wird eine extra Zeitspalte im Ergebnis erzeugt die die angegebenen Zeitstampel anzeigt.

**<Not>**

Nots erben von StructureContainingElem. Sie haben jedoch immer nur genau ein Child-Element. Alle Bedingungen des enthaltenen Kindes dürfen nicht erfüllt sein damit Ergebnisse im Gesamtergebnis auftauchen.

**<SubQuery>**

SubQueries sind gespeicherte Suchen die in anderen Suchen integriert werden können. Vor der Ausführung einer Suche wird die SubQuery durch ihren eigentlichen Inhalt ausgetauscht so dass für einen Suchinterpreter dieses Element niemals auftaucht.

Parameter:

optional:

 Datentyp: Boolean Default: FALSE

 Bedeutung: wenn auf TRUE werden in der expandierten Subquery bei alle Elemente das Flag „optional“ auf TRUE gesetzt.

displayColumns:

 Datentyp: Boolean Default: TRUE

 Bedeutung: wenn auf FALSE werden in der expandierten SubQuery bei allen Attributen die Flags „displayValue“, „caseID“, „docID“ und „infoDate“ auf FALSE gesetzt. Über diesen Mechanismus kann erzwungen werden dass die SubQuery keine eigenen Spalten im Ergebnis erzeugt

name:

 Datentyp: String

Bedeutung: Der Name inklusive des Pfades der Query die an Stelle dieses Platzhalters eingesetzt werden soll

queryID:

 Datentyp: Integer

Bedeutung: Die ID der Query die an diese Stelle eingesetzt werden soll. Der Parameter QueryID hat vorrang vor dem Parameter Name. Auf diese Art können die SubQueries umbenannt oder verschoben werden, ohne dass andere Queries die diese verwenden kaputt gehen.

**<DistributionColumn>**

DistributionColumns erben von StructureContainingElem. Sobald ein solches Element in einer Suche auftaucht gilt die gesamte Query als Verteilungssuche. Für eine Verteilungssuche werden mehrere Suchen durchgeführt die jeweils mit einem Element aus dem in der QueryDistributionColumns enthaltenen Elemente konfiguriert ist.

**<DistributionRow>**

DistributionRow erben von StructureContainingElem. Siehe DistributionColumn.

**<DistributionFilter>**

DistributionFilter erben von StructureContainingElem. Siehe DistributionColumn.

**<SortOperator>**

 <Sort attrID= elementID sortDir=ASC|DESC sortOp=Value|Timestamp|PID|CaseID >

**Changes:**

**21.10.2016**

- Entfernung von „selectString“ aus Query

- Entfernung von „desiredContentBetweenUpperBound“ aus QueryAttribute. Dieser Member wird in Zukunft in den „desiredContent“ hineinkodiert

- Entfernung von „internalID“ aus QueryElem

- Entfernung von „alias“ aus QueryAttribute

- Ergänzung von „elementID“ zu QueryAttribute

- Ergänzung des QueryAttribute-Members “minimumCount”.

- Ergänzungen zum „desiredContent“ von QueryAttribute

- Erhöhung der Version auf 0.2

**26.10.2016**

- Ergänzung von „CONTAINS\_NOT“ und „CONTAINS\_NOT\_POSITIVE“ in den ContentOperatoren von queryAttribute

- Erhöhung der Version auf 0.3

**31.10.2016**

- Ergänzung von “minCount”, “maxCount”, “countType” bei QueryAttribute

- Erhöhung der Version auf 0.4

**2.12.2016**

- QueryRoot und IDFilter liefern bei distinct das jeweils kleinste Element au seiner möglichen Menge von Fakten zurück.

- Erhöhung der Version auf 0.5

**21.12.2016**

- Ergänzung von ExtractionMode zu QueryAttribute

- Erhöhung der Version auf 0.6

**9.2.2017**

- Queries mit einer Version unter 0.7 die keinen IDFilter-Case beinhalten werden von der Solr-Engine so interpretiert als ob sie einen Case-Filter besitzen würden. Mit der neuen Möglichkeit eine Patienten-Ebene zu indexieren wird dies ab jetzt nicht mehr gemacht, sondern diese Queries werden als echte Patient-Queries interpretiert.

**-** Erhöhung der Version auf 0.7

**16.5.2017**

- Subquery bekommt die Attribute „optional“ und „displayColumns“

- Erhöhung der Version auf 0.8