## Соответствия элементов языка AQL и объектной модели (языка СИНТЕЗ)

|  |  |
| --- | --- |
| **Элемент AQL DDL** | **Элемент объектной модели** |
| Database | ModuleDef |
| Database.name | ModuleDef.name |
| Database.arrays | ModuleDef.classes |
| Array | ClassDef, ADTDef |
| Array.name | ClassDef.name |
| Array.attributes | ClassDef.instanceType.attributes |
| Array.dimensions | ClassDef.instanceType.attributes |
| Attribute | AttributeDef |
| Attribute.name | AttributeDef.name |
| Attribute.type | AttributeDef.type |
| Dimension | AttributeDef |
| Dimension.name | AttributeDef.name |
| Dimension.type | AttributeDef.type |
| string\_aql | StringDef |
| bool\_aql | BooleanDef |
| datetime\_aql | TimeDef |
| double\_aql | RealDef |
| float\_aql | RealDef |
| int8\_aql | IntegerDef |
| int16\_aql | IntegerDef |
| int32\_aql | IntegerDef |
| int64\_aql | IntegerDef |
| uint8\_aql | IntegerDef |
| uint16\_aql | IntegerDef |
| uint32\_aql | IntegerDef |
| uint64\_aql | IntegerDef |

|  |  |
| --- | --- |
| **Элемент ЯМД объектной модели** | **Элемент AQL DML** |
| Rule, Atom | SelectAttributes |
| Rule.head.terms->type.elements,  Atom.terms->type.elements | SelectAttributes.attributes |
| Rule.body.formula | SelectAttributes.where,  SelectAttributes.fromClause,  SelectAttributes.join,  SelectAttributes.joinOn |
| Atom.symbol | SelectAttributes.fromClause(Path)->components |
| ReductElementDef, Variable | Path |
| ReductElementDef.path->attribute.name,  Variable.name | Path.components |
| ReductElementDef.name | Path.alias |
| FunctionCall | FunctionExp |
| FunctionCall.name | FunctionExp.name |
| FunctionCall.terms | FunctionExp.arguments |
| MultiplicativeOpCall, AdditiveOpCall, RelationPredicate | BinaryExp |
| MultiplicativeOpCall.name, AdditiveOpCall.name,  RelationPredicate.symbol | BinaryExp.symbol |
| MultiplicativeOpCall.terms, AdditiveOpCall.terms,  RelationPredicate.terms | BinaryExp.leftExp, BinaryExp.rightExp |
| BracketValue | BracketExp |
| BracketValue.terms | BracketExp.exp |
| StringValueDef | StringValue |
| StringValueDef.value | StringValue.value |
| RealValueDef | FloatValue |
| RealValueDef.value | FloatValue.value |
| IntValueDef | IntegerValue |
| IntValueDef.value | IntegerValue.value |
| BooleanValueDef | BooleanValue |
| BooleanValueDef.value | BooleanValue.value |

## Отображение языка запросов Syfs в AQL

* в голове правила – предикат-коллекция
* тело – конъюнкция предикатов-коллекций и арифметических предикатов
* в предикате-коллекции переменная типизирована редуктом
* условия для соединения – только равенства атрибутов вида a = b. Все атрибуты – разные (возможно, с помощью переименования)
* выражения в редукте – только переименования атрибутов a: b, без путей, в теле правила
* термы – имена атрибутов, константы, арифметические операции, вызовы функций
* вызовы функций – только как термы, не как часть пути
* В трансформацию должен передаваться список уникальных атрибутов (измерений). Те измерения, которые есть – не включать в AQL-select (они там есть по умолчанию). Неясно, что делать, если в запросе перечислены не все измерения – AQL все равно вернет все (нельзя просто отрезать измерение). Поэтому нужно считать что в голове всегда перечислены все измерения.
* Все сущности (типы, классы, функции), используемые в запросе, собирается в модуль *Query*, сам запрос добавляется в качестве реализации функции *query*.
* переименование атрибутов производится через alias (as) в select
* если соединение коллекций преобразуется в JOIN, в секцию ON попадают условия на равенство атрибутов

*Замечание.* Соединение в AQL в FROM – это inner join (merge ?), измерения у массивов должны совпадать, атрибуты соединяются. Соединение в JOIN-ON – это cross join, количество измерений соединения - #dim(array1) + #dim(array2) - #dim(on).

## Синтаксис языка запросов объектной модели

<value> ::= <attribute identifier> | <typed variable> | <boolean value> | <number> | <string>

<rule> ::= <atom> :- <body>.

<body> ::= <formula>

<formula> ::= <atomic formula> | <formula> & <formula>

<atomic formula> ::= <atom> | <term> <relation predicate> <term>

<atom> ::= <collection name>(<typed variable>)

<typed variable> ::= <variable identifier> / <type expression>

<type expression> ::= <type identifier> ‘[’ <reduct element list> ‘]’

<reduct element> ::= [<rename identifier> :] <attribute identifier>

<term> ::= <arithmetic expression>

<arithmetic expression>::= <value> | <function designator> | (<arithmetic expression>) |

<arithmetic expression> <arithmetic operation> <arithmetic expression>

<arithmetic operation> ::= + | - |\* | /

<function designator> ::= <function name>(<term list>)