In our understanding, the user will not need the full power of the query language to construct their criteria. Therefore, we are looking for a compromise that would reconcile the following features of the structure of the query conditions:

* sufficient possibilities of expression,
* conceptual simplicity,
* a form convenient for restoring the saved condition and editing it,
* a convenient form for passing the condition for evaluation on the server.



The proposed conceptual model for the conditions defined inside the filter is shown above. It is shown in the form of an object model, although eventually it will take the form of an XML structure and a relation passed through it, which will become a parameter of the appropriate SQL procedure.

The abstract *Predicate* class represents any (sub)expression that returns a Boolean value. Its contextEntity attribute tells you the set of what kind of entity is filtered by such an expression.

Such predicates can be combined using logical operators. The way they are mapped in the model will ensure that the order given to them by the user in the editor is preserved.

The *SimplePredicate class*  represents an expression composed of the feature ofan operator and a (perhaps optional) value (for example, comparison operators).

AggregatePredicate, on the other hand, is based on the assumption that only (actually stored or enumeratable) properties belonging to a given entity will be able to appear inside aggregate operations (identified by the aggregateFunction attribute). The aggregatedEntity attribute points to the entity, the set of instances whose instance it is used to enumerate the aggregation. I.e. for the query Performer where count(Session)>10, contextEntity = 'performer' and aggregatedEntity='session' .

A PredicateGroup represents a parenthetically extracted subexpression. In the enclosing chain of related subexpressions, the PredicateGroup object will appear as "next" for the last expression before its opening parenthesis (if there is one), and as "previous" for the next subexpression after its closing parenthesis (if there is one).