Руководство по разработке фильтров Communiware версия 0.912

В.Б. Вагнер

11 мая 2004 г.

Оглавление

Глава 1

Понятие фильтра

Фильтры — одно из основополагающих понятий в Communiware. Фильтр является единственным способом породить список каких-либо сущностей (обычно итемов). Кроме того, фильтр является основным способом создания контекста итема, отличного указанного в URL.

Фильтры используются для создания списков, в элементах форм для создания выпадающих меню и групп кнопок. Кроме того, фильтры могут быть использованы в логических выражениях для проверки принадлежности объекта какой-либо группе.

1.1 Виды фильтров

Поскольку в основе Communiware лежит SQL-база данных, наиболее распространенным видом фильтров являются SQL-запросы. В некоторых случаях, когда возможностей SQL не хватает, например, надо работать с объектами файловой системы или просто условие отбора зависит от параметров контекста слишком сложным образом, чтобы его можно было выразить в виде одного SQL-запроса, используются фильтры, написанные на Perl.

В некоторых случаях использование SQL- или Perl-фильтров, наоборот является излишним усложнением задачи, поскольку набор объектов известен заранее. Для этих случаев предусмотрен синтаксис с явным перечислением объектов.

1.1.1 SQL-фильтры

SQL-фильтр представляет собой параметризованный SQL-запрос, которому присвоено некоторое имя. Он создается через специализированный Web-интерфейс, и хранится в базе данных как отдельная сущность.

Наряду с типами итем и типами связей, SQL-фильтры рассматриваются как часть онтологии, и переносятся с сервера на сервер с помощью XML-файлов удовлетворяющих Ontology DTD.

Для указания мест подстановки параметров в фильтр используется синтаксис, принятый в DBI — стандартном интерфейсе доступа к базам данных языка Perl. В тексте SQL-запроса константа, роль которой должен играть параметр, заменяется вопросительным знаком, и при выполнении фильтра ему передеатся количество значений, в точности соответствующее количеству вопросительных знаков в тексте запроса.

Подставлять таким способом можно только константные значения используемые в SQL-выражениях, но не имена таблиц и полей, и тем более, не ключевые слова SQL.

Такая система не совсем удобна, так как достаточно часто требуется подставить одно и то же значение в текст SQL-запроса несколько раз. Кроме того, фильтры, как правило, работают не с явно указанными значениями, а с именованными атрибутами контекста Communiware (подробнее об этом рассказывается в главе 2.

Поэтому, кроме собственно текста SQL-запроса в состав фильтра входит список аргументов, представляющий собой перечисление имен атрибутов контекста, соответствующих вопросительным знакам в тексте запроса.

Синтаксис вызова SQL-фильтра в шаблоне представляет собой просто его имя, за которым, возможно, в скобках следуют *позиционные* параметры (см. раздел 2.2.1).

1.1.2 Perl-фильтры

Фильтры, написанные на Perl представляют собой часть программного кода Communiware. Создание их через Web-интерфейс невозможно, так как они предоставляют слишком большие возможности доступа к операционной системе сервера. Для установки нового Perl-фильтра требуется иметь доступ к командной строке сервера с правами привелигированного пользователя.

Perl-фильтр представляет собой модуль языка Perl, определяющий

объект, обычно наследник от `Communiware::Filter`. Имя модуля должно быть в пространстве имен `Communiware::Filter`, например `Communiware::Filter::Pic`.

При использовании фильтра в шаблоне общаяя часть имени не указывается. Например, на вышеупомянутый фильтр ссылкются как `::Pic`.

Имя фильтра, начинающееся с двух двоеточий служит указанием для интерпретатора шаблонов, что следует искать не SQL-фильтр, а модуль Perl.

Объект-фильтр должен иметь конструктор `new`, создающий объект, которому в качестве параметров передаются все параметры фильтра, указанные в шаблоне, и поддерживать методы `fetchrow_hashref`, `fetchall_arrayref` и `finish`, аналогичные соответствующим методам хэндлов SQL-операторов DBI.

Кроме того, в модуле должна быть определена функция `check`, **не являющаяся методом объекта**, которая получает те же параметры, что и конструктор `new`, и возвращает любое истиное значение, если параметры корректны, и **`undef`**, если параметры указаны с ошибками.

От методов фильтра не требуется полной функциональности соответствующих методов DBI. Достаточно, если `fetchrow_hashref` будет при каждом вызове возвращать ссылку на хэш с атрибутами очередного элемента списка, причем имена атрибутов должны быть в верхнем регистре, а `fetchall_arrayref` — ссылку на массив ссылок на хэши со всеми еще не извлеченными элементами.

Внутри реализации фильтра можно пользоваться функциями модуля `Communiware::Context` для доступа к атрибутам контекста и вывода информации на генерируемую страницу, а также методами объекта `$Communiware::dbh` для прямого доступа к базе данных.

1.1.3 Явное указание списка итем

Синтаксис фильтра, содержащего явное перечисление итем очень прост — в круглых скобках через запятую список объектов.

Созданные этим фильтром контексты будут содержать только атрибут `ITEM_ID`, в который будут последовательно помещаться указанные в списке значения.

Поскольку при подстановке в параметр динамического элемента значений типа список (например, результата выбора из нескольких одноименных чекбоксов) получается список через запятую, то для органи-

зации перебора выделенных пользователем значений можно воспользоваться конструкцией вида

```
<:Loop (@CHECKBOX):>
....
<:EndLoop:>
```

1.2 Способы использования фильтров

1.2.1 Списки

Построение списков итемов с помощью динамических элементов Loop и List — наиболее распространенный способ использования фильтров в шаблонах Communiware.

Эти динамические элементы содержат шаблон фрагмента страницы (Loop — непосредственно, для List шаблон указывается по имени), который обрабатывается в контексте каждого элемента, возвращенного фильтром.

Как правило, фильтры, используемые в List и Loop возвращают список итемов. Это означает, что среди атрибутов, возвращенных фильтром, есть атрибут с именем `ITEM_ID`. Если в контексте присутствует идентификатор итема, то все остальные атрибуты этого итема доступны для Communiware. При необходимости, для того чтобы их извлечь из базы будет автоматически выполнен дополнительный SQL-запрос. Поэтому нет необходимости стремиться к тому, чтобы фильтр самостоятельно извлекал все атрибуты, которые потребуются в шаблоне.

Фильтры используемые в динамических элементах Loop и List подвергаются обработке *машиной переписывания запросов*, подробно описанной в главе 5. Поэтому в большинстве случаев необходимые в шаблоне атрибуты будут инкорпорированы в запрос автоматически.

Также автоматически будут учтены критерии сортировки, заданные в Loop или List.

В некоторых случаях используются списки объектов, которые не являются итемами, например, список иллюстраций текущего итема, или список его атрибутов. В этом случае фильтр порождает контексты, не содержащие `ITEM_ID` и машина переписывания к таким фильтрам не применяется.

Тем не менее, результат такого фильтра может быть отсортирован с помощью обычных средств сортировки списков.

1.2.2 Логические выражения

В синтаксисе логических выражений Communiware предусмотрена операция **IN**, проверяющая значение на вхождение в результат фильтра. Эта операция всегда подразумевает, что проверяемое значение является идентификатором итема, и проверяет его на совпадение со значением атрибута `ITEM_ID`, возвращенного фильтром.

Близим по смыслу к этой логической операции является использование фильтров в динамическом элементе Authentify. Здесь проверяется попадание идентификатора текущего пользователя (который с очевидностью является идентификатором итема) в результат, возвращаемый фильтром.

1.2.3 Фильтры подписки

Несколько особую роль играют фильтры в системе почтовых рассылок. Здесь результат выполнения фильтра (вернее, наличие или отсутствие извлеченных итемов) позволяет решить, есть ли вообще необходимость обрабатывать шаблон и посылать письмо. Поэтому фильтр подписки указывается не в тексте шаблона, как один из параметров динамического элемента, а как значение атрибута шаблона, чтобы он был доступен демону рассылки до начала обработки текста шаблона.

Кроме того, фильтры подписки должны учитывать два атрибута контекста — `PERIOD_START` и `PERIOD_END`. Эти атрибуты существуют только в контексте рассылки и задают временной интервал в который должны попадать итемы, рассылаемые в данный момент. Обычно, на попадание в этот интервал проверяется дата публикации итема, хотя возможны ситуации, когда большее значение имеет дата последнего изменения (`LASTCHANGE`) или дата последнего изменения статуса (`STATUS_UPDATE`).

Для того, чтобы избежать повторного выполнения фильтра подписки и проверок прав доступа, производимых над его результатом, внутри шаблона рассылки используется специальный Perl-фильтр ::Mail, использующий сохраненный демоном подписки результат.

1.2.4 Динамические элементы MailTo и Subscribe

В динамическом элементе MailTo фильтр используется для формирования списка адресатов. Поскольку все адресаты являются пользовате-

лями Communiware, этот фильтр должен возвращать контексты итемов типа «Персона». Как правило, атрибут EMAIL, используемый для рассылки, извлекается средствами машины переписывания запросов.

В динамическом элементе Subscribe фильтр возвращает список подписчиков. Здесь нас интересуют только их идентификаторы, а не адреса, поскольку в при хранении подписки в базе данных запоминаются только идентификаторы.

1.2.5 Динамический элемент Input

В динамическом элементе Input фильтры используются для формирования конструкций выбора. Хотя и здесь, например при задании связей в формах постинга, выбираемыми объектами достаточно часто являются итемы, Input-элементы — самый распространенный случай в Communiware, когда фильтр возвращает не итемы.

Фильтр, используемый в динамическом элементе Input должен возвращать по крайней мере два атрибута — значение и его текстуальное представление, которое выводится на экран. Поэтому в Input-элементах практически никогда не используются фильтры с явным указанием списка итемов. Вместо этого предусмотрен специальный синтаксис для явного указания позиций меню.

1.3 Интерфейс редактирования фильтров

Для редактирования фильтров в Communiware предусмотрен специальный Web-интерфейс. Войти в него можно, выбрав «Фильтры» в меню «Прочие объекты» в интерфейсе вебмастера. При этом на экране появляется список SQL-фильтров, существющих на сервере. Выбрав в этом списке ссылку «Создать новый фильтр» или войдя в просмотр нужного фильтра, и выбрав ссылку «Редактировать» мы попадаем в интерфейс редактирования фильтров.

В форме редактирования фильтров присутствует поле для ввода имени фильтра. Изменив имя существующего фильтра в этом поле, и нажав кнопку «Сохранить» можно создать новый фильтр. Это достаточно распространенная практика — создавать новые фильтры путем небольших изменений существующих.

В верхней части формы присутствует список шаблонов, в которых используется данный фильтр. Если этот список пуст, появляется кнопка

«Удалить».

Ниже расположено поле для ввода SQL-запроса. Под ним расположено поле «Аргументы», куда вводятся имена атрибутов контекста, используемых данным фильтром (см. главу 2).

Ниже расположен чекбокс «Разрешить переписывание». Сняв отметку в нем можно запретить обработку фильтра машиной переписывания (см. главу 5.

Под этим полем расположены поля «Значения для тестирования» и «Атрибуты для извлечения», предназначенные для отладки фильтров.

В поле «Значения для тестирования» вводятся значения, соответствующие атрибутам контекста, упомянутым в строке «Аргументы». Они должны быть в том же порядке, т.е. каждому вопросительному знаку в тексте запроса должно соответствовать значение в этом поле. Значения разделяются запятыми. Значения типа *dama* следует вводить в стандартном формате Communiware YYYY.MM.DD HH.MI.SS.

Рядом с этим полем расположена кнопка «Тест» позволяющая выполнить фильтр с указанными значениями и получить его результаты.

При этом фильтр не сохраняется в базе, что позволяет отлаживать модифицированный фильтр на работающем сайте. Пока ведется работа с фильтром, шаблоны продолжают использовать старую версию.

Поле «Атрибуты для извлечения» предназначено для тестирования переписывания фильтра. Туда вводится список атрибутов итема, которые должны извлекаться фильтром. Если перед именем атрибута указывается знак «+» или «-» это означает, что результаты фильтра должны быть отсортированы по значениям этого атрибута, соответственно, в порядке возрастания или убывания.

В случае, если это поле не пусто, при нажатии кнопки «Тест» или «План выполнения» помимо собственно результатов операции выводится текст переписанного SQL-запроса.

Кнопка «План выполнения» позволяет получить план выполнения запроса, т.е. перечнь операций которые будет выполять SQL-сервер при выполнении данного запроса. Анализ плана выполнения и оптимизация запроса с учетом этого анализа позволяет иногда ускорить время выполнения фильтра на порядок. Подробнее об этом см. в разделе 7.2.

Последним полем ввода в этой форме является поле для описания фильтра. В этом поле рекомендуется описывать в нескольких словах назначение фильтра и перечислять смысл всех позиционных параметров данного фильтра, поскольку понять это по тексту SQL-запроса не всегда легко.

Непосредственно под этим полем расположена кнопка «Сохранить», записывающиая отредактированный фильтр в базу.

Глава 2

Взаимодействие фильтров с контекстом

2.1 Использование фильтром атрибутов контекста

Большая часть фильтров отталкивается от текущего контекста. Нам практически никогда не нужен список „итемов вообще” — нас интересуют, например, „статьи в текущей рубрике” или „реплики на текущую статью”. Даже в том случае, когда нас интересуют „все статьи текущего сайта”, нам нужно откуда-то получить информацию о том, какой сайт является текущим, не говоря уже о том, что как правило подобные списки обычно строятся в контексте итема сайта.

Поэтому, фильтры обычно используют в качестве аргументов атрибуты контекста. Обычно в их число входит `ITEM_ID`, достаточно часто `CURRENT_SERVER`. Для того, чтобы указать, что некоторому вопросительному знаку в тексте запроса должен быть поставлен в соответствие атрибут контекста, достаточно просто указать имя этого атрибута в соответствующем месте списка аргументов фильтра.

В списке аргументов один и тот же атрибут контекста может быть упомянут несколько раз, в тех случаях когда удобно в тексте SQL-запроса несколько раз упомянуть одно и то же значение. (Особенно это касается запросов с UNION и другими множественными операциями).

2.1.1 Использование атрибутов типа дата

Атрибуты типа *строка* и *число* корректно подставляются в запросы без всяких усилий со стороны разработчика фильтров. Несколько сложнее ситуация с атрибутами типа *дата*. В Communiware эти атрибуты хранятся в формате YYYY.MM.DD HH.MI.SS, который не воспринимается как дата большинством SQL-серверов. Поэтому необходимо явно преобразовать аргумент в значение типа *дата* с помощью хранимой функции `tdate`. Например:

```
select item_id from item where PUBLISHED>tdate(?)
```

2.2 Параметры фильтров

Если бы все значения, от которых зависит результат фильтра, не считая атрибутов контекста, приходилось указывать в тексте запроса в явном виде, пришлось бы создавать очень много похожих друг на друга фильтров — один для отбора потомков по связи TOPIC, другой — для отбора потомков по связи PART и так далее. Поэтому в Communiware предусмотрен механизм параметризации фильтров.

Параметры фильтра указываются в шаблоне в скобках после имени фильтра.

2.2.1 Позиционные параметры

Наиболее распространенным случаем параметров фильтра являются позиционные параметры. Т.е. при разработке фильтра принимается решение, что первый по счету параметр фильтра задает, например, тип связи, второй — тип итема и так далее.

В строке аргументов на позиционные параметры ссылаются по их номерам 1, 2 и так далее.

Можно считать, что фильтр с параметрами выполняется в расширенном контексте, в который добавлены атрибуты с числовыми именами, значениями которых являются параметры, переданные фильтру.

2.2.2 Множественные параметры

В некоторых случаях необходимо создавать фильтры, которые получают список параметров заранее неизвестной длины. Например, фильтр PassiveMultiTyped возвращает список потомков по некоторой связи,

имеющих типы из определенного списка. Этот список задается при разработке шаблона и может содержать от двух до шести значений.

Аналогичным способом устроены и другие фильтры с переменным числом параметров — среди этих параметров есть список однородных величин, длина которого заранее неизвестна.

В тексте запроса этому списку соответствует конструкция

имя-поля `in (?,?,?,?,?)`

а в строке аргументов — звездочка. В строке аргументов может присутствовать ровно одна звездочка, которой будут соответствовать все параметры, кроме нескольких первых, явным образом указанных среди аргументов. Иначе будет непонятно, какие из аргументов следует использовать для первого списка, а какие для второго.

Длина списка переменных параметров должна быть не менее 1 и не более числа вопросительных знаков, использованных разработчиком фильтра в соответствующей конструкции `in`.

Следует учитывать, что фильтры с большим числом аргументов исполняются менее эффективно, поэтому не рекомендуется при разработке фильтров резервировать более 4–6 позиций на список переменных параметров.

2.3 Контексты создаваемые фильтром

Результатом работы фильтра всегда является послдовательность контекстов. Контекст это набор именованных атрибутов.

В случае SQL-фильтра имена атрибутов, возвращенных запросом, совпадают с именами колонок возвращенных SQL-запросом. Поэтому в случаях, когда колонка в результате запроса имеет не тот смысл, который она должна иметь в контексте (например, мы хотим вернуть значения поля passive таблицы item_link как `ITEM_ID`, необходимо использовать выражение AS. Например:

```
select passive as ITEM_ID from item_link where
active=? and linktype_id=? order by ordnum
```

Все параметры типа *дата*, возвращаемые фильтром, должны иметь стандартный внутренний формат Communiware: YYYY.MM.DD HH.MI.SS.

Поскольку в большинстве SQL-серверов формат даты по умолчанию иной, Communiware определяет хранимую функцию `fdate`, которой *обязательно* надо пользоваться при извлечении параметров типа *дата*. Поскольку в этом случае имя поля не будет совпадать с именем поля-

аргуменра fdate, использование конструкции AS с полями типа *дата* обязательно.

Например:

```
select item_id, fdate(LASTCHANGE) as LASTCHANGE from
item where ...
```

2.3.1 Контексты итема и контексты, не содержащие ITEM_ID

Контексты, создаваемые фильтрами, можно разделить на две большие группы — контексты итемов и контексты, не содержащие ITEM_ID.

Разница между этими двумя типами контекстов принципиальная — если в контексте присутствует атрибут ITEM_ID, значением которого является идентификатор существующего итема, механизм контекстов Communiware позволяет обращаться ко всем атрибутам этого итема, независимо от того, были ли они помещены в контекст фильтром или нет. В контекстах итемов возможно также вычисление групповых и вычислимых атрибутов.

В контексте, не содержащем ITEM_ID доступны только те атрибуты, которые были непосредственно возвращены фильтром. Область применения таких контекстов крайне ограничена. Иногда формируются списки из таких контекстов, например список иллюстраций итема, или список его атрибутов.

Довольно часто, такие контексты используются для формирования меню и списков выбора.

2.3.2 Явное извлечение атрибутов итема фильтром

Разработчик фильтра может перечислить в предложении select SQL-запроса разнообразные атрибуты итема, кроме ITEM_ID. В этом случае, они будут извлечены фильтром, даже если тот не будет обработан машиной переписывания.

Особенно полезно извлекать дополнительные атрибуты в случае *непереписываемых* фильтров (см главу 5), поскольку расходы на возвращение нескольких лишних колонок при выполнении фильтра очень малы по сравнению с выполнением лишнего SQL-запроса на каждый элемент списка. При написании непереписываемых фильтров следует помнить, что помимо атрибутов явным образом использованных в ша-

блоне, Communiware *всегда* использует атрибут LASTCHANGE, необходимый для формирования HTTP-заголовка Last-Modified.

Принципиально, можно написать фильтр, который, скажем в колонке TITLE будет возвращать нечто отличное от заголовка того итема, идентификатор которого возвращается в колонке ITEM_ID. Результат подобных действий вообще говоря непредсказуем. Может быть вам и удастся воспользоваться этим значением, а может быть в результате обращения к какому-либо другому атрибуту данного итема, набор его атрибутов будет перечитан из базы, и в атрибуте TITLE окажется настоящий заголовок этого итема.

Поэтому рекомендуется *никогда* не использовать имен, уже занятых атрибутами итема для чего-либо, кроме значений этих атрибутов.

2.3.3 Переопределение групповых и вычислимых атрибутов

Совершенно по другому обстоит дело с групповыми и вычислимыми атрибутами. В отличие от атрибутов итема, которые механизм контекстов извлекает группами (все атрибуты, хранящиеся в одной таблице — вместе), каждому групповому или вычислимому атрибуту соответствует отдельный SQL-запрос. Поэтому, если вам удается вычислить в SQL-запросе значение, эквивалентное значению группового или вычислимого атрибута, лучше его назвать тем именем, которое бы это значение получило будучи вычисленным отдельно. Это облегчит разработчикам использование контекста, созданного вашим фильтром — они будут использовать один и тот же синтаксис, независимо от того, был ли атрибут вычислен фильтром или нет, а шаблон в котором число SQL-запросов существенно уменьшено за счет того, что групповые атрибуты для всего списка посчитаны разом, будет выполняться эффективнее.

2.3.4 Создание специфичных для фильтра атрибутов

В случае, если необходимо вычислить какое-то значение для каждого элемента списка, а затем использовать его в шаблоне, можно написать SQL-фильтр, который кроме ITEM_ID и каких-либо атрибутов итема возвращает колонку с уникальным именем. Значения этой колонки будут помещены в контекст и могут быть использованы как атрибуты.

Следует учитывать, что атрибуты Communiware имеют тип, который учитывается многими динамическими элементами, а SQL-фильтры

не передают в контекст информацию о типе возвращаемых атрибутов. Поэтому по умолчанию, все атрибуты, созданные фильтром будут рассматриваться как строки.

Разработчик Perl-фильтра может явным образом специфицировать типы возвращаемых атрибутов. Для этого необходимо в функции check создать элементы глобального хэша `$Communiware::LocalType` ключами которых являются имена возвращаемых атрибутов, а значениями — их типы. Например, фильтр `::Pic` определяет свои атрибуты следующим образом:

```
$Communiware::LocalType{'PICTURE'}='STRING';
$Communiware::LocalType{'WIDTH'}='NUMBER';
$Communiware::LocalType{'HEIGHT'}='NUMBER';
$Communiware::LocalType{'FILESIZE'}='NUMBER';
```

Разработчик SQL-фильтра такой возможности лишен. Поэтому, при необходимости явным образом задать тип специфического для фильтра атрибутов как *число* или *дату*, нужно использовать *суффиксную нотацию*. Communiware считает, что все атрибуты, имена которых кончаются на _N, числовыми, на _D — датами, и на _R — содержащими форматированный текст (RICHTEXT). Поэтому, если вам необходимо, чтобы возвращаемый фильтром атрибут, который по смыслу должен называться COUNT, воспринимался как числовой, назовите его COUNT_N.

Естественно, что значения атрибутов типа *дата* должны быть сформированы с помощью функции fdate.

Глава 3

Унифильтры

Начиная с версии 0.93 в Communiware появился еще один тип фильтров — так называемые «унифицированные фильтры» или «унифильтры». Как и другие типы фильтров в Communiware унифильтры предназначены для формирования множеств объкетов. Ближе всего они к „обычным” SQL-фильтрам, но имеют следующие основные отличия от них.

- Унифильтры предназначены для отбора только итемов, они не могут формировать множества каких-либо других объектов Communiware, например графических файлов или типов связей.

- Текст унифильтра записывается на специальном языке, оперирующем понятиями графовой структуры Communiware (типы связей, атрибуты и т.п.), и не требующем знаний языка SQL и структуры таблиц Communiware.

- Текст унифильтра может храниться в базе данных (так же, как и для SQL-фильтров). Но кроме этого возможно явное указание текста унифильтра непосредственно в тексте шаблона.

- Унифильтры позволяют выполнять „уточнение” условий. Т.е. можно выразить критерий отбора вроде „хочу получить все итемы, отбираемые указанным хранимым унифильтром, но удовлетворяющие, кроме того следующим, явно заданным условиям”.

3.1 Синтаксис языка описания унифильтров

Текст унифильтра представляет собой ноль или больше *условий* разделенных запятыми и заключенных в фигурные скобки - { и }. Вот два примера синтаксически правильных унифильтров:

```
{}
{TYPE_ID = 'TOPIC', (PART, TOPIC) <--}
```

Первый из них не содержит никаких условий вообще (что допустимо), а второй - два условия: `TYPE_ID = 'TOPIC'` и `(PART, TOPIC) <--`. Унифильтр отбирает итемы, удовлетворяющие *всем* приведенным условиям, т.е. условия объединяются по логическому И.

3.1.1 Атрибуты контекста и атрибуты итемов

Напомним некоторые понятия, употребимые в Communiware. Как и любой другой фильтр унифильтр выполняется в определенном контексте, т.е. множестве поименованных и имеющих значения *атрибутов контекста*. Напомним, что контекст определен во время выполнения фильтра, и может изменяться со временем. В отличие от атрибутов контекста *атрибуты итема* - это поименованные значения связанные с конкретным итемом. В текущем контексте могут отсутствовать какие-то из используемых унифильтром атрибутов. В этом случае они считаются неопределенными. Понятия атрибутов контекста и атрибутов итемов активно используются при описании различных типов условий, возможных в унифильтрах.

3.1.2 Условия отношения

К этой группе условий относятся условия, накладываемые на значеня атрибутов отбираемых итемов: „дата пуликации не ранее чем . . . ”, „адрес электронной почты равен . . . ” и т.п. Условия отношения имеют один из двух вариантов синтаксиса:

```
<атрибут итема> <операция> <значение>
<атрибут итема> <операция> (<значение>, <значение>,...)
```

Операция - это

= – равенство,

`!=` или `<>` – неравенство,

`>` – больше,

`<` – меньше,

`>=` – больше или равно,

`<=` – меньше или равно,

`like` – операция текстовой „похожести”, такая же, как в языке SQL.

Как видно в правой части условия может находиться либо единичное значение, либо список значений, разделенных запятыми. В этом случае условие трактуется, как логическое `ИЛИ` для всех значений из списка. Условие `ITEM_ID = ('aaa', 'bbb', 'ccc')` означает, что атрибут ITEM_ID в искомом множестве итемов должен быть равен или 'aaa', или 'bbb' или 'ccc'.

Значение в правой части условий отношения может иметь следующие варианты.

`Название атрибута контекста` - действительно, название атрибута контекста, например `ITEM_ID, PUBLISHED, SYSDATE`.

`Числовой литерал` - последовательность цифр 0–9: `1, 456, 789001`.

`Строковый литерал` - последовательность произвольных символов, заключенная в апострофы: `'пример строкового литерала'`.

`Литерал типа дата` - предназначен для указания в тексте унифильтров константных дат, имеет вид 'YYYY.MM.DD HH.mm.SS'. Обязательно указание всех частей даты, до секунд включительно. Пример литерала типа дата: `'2001.04.01 12.24.37'`.

`Выражение над датой` - в качестве значений в условиях язык описания унифильтров допускает простые выражения над датами имеющие вид `<название атрибута контекста> [+-] <целое число>`. Таким образом можно выразить „дату на 10 дней большую, чем. . . ”. Пример условия с выражением над датой: `PUBLISHED > SYSDATE - 10`.

Несколько примеров унифильтров, использующих условия отношения.

`{}`
: „Минимальный" унифильтр. Так как никаких ограничивающих условий не указано, то он отбирает *все* итемы, имеющиеся в момент его выполнения.

`{TYPE_ID = 'TOPIC'}`
: Отбираются все итемы типа TOPIC.

`{TYPE_ID = ('TOPIC', 'PART')}`
: Отбираются все итемы типа TOPIC или типа PART.

`{ITEM_ID like '%test%'}`
: Отбираются итемы, идентификатор которых содержит в себе строку `test`, например `testname, svrtest5`...

`{SERVER = 'default', PUBLISHED > SYSDATE -10}`
: Отбираются итемы принадлежащие серверу `default` и опубликованные в последние десять дней.

`{TITLE > TITLE}`
: Хотя и не очень осмысленный, но вполне работоспособный фильтр. Отбираются итемы, у которых значение атрибута TITLE (левая часть условия) больше, чем значение атрибута TITLE текущего контекста (правая часть).

Важное замечание относительно условий отношения с атрибутами контекста в правой части. Если при выполнении унифильтра атрибут контекста, входящий в правую часть условия не определен, то все это условие „редуцируется" и не оказывает никакого влияния на результат работы унифильтра. Например, если при работе последнего из приведенных выше примеров не определен атрибут контекста TITLE, то фильтр возвратит множество всех итемов, как если бы его текст был `{}`.

3.1.3 Условия связи

Описанные выше условия отношения задают требования к значениям атрибутов итема. *Условия связи* позволяют выразить требования к достижимости одних итемов из других по определенным типам связей. Синтаксис условий связи следующий:

`<типы связей> <операция> <идентификаторы итемов>`

<*Типы связей*> - это или название одиночного типа связи, или перечисление через запятую нескольких типов связей, заключенных в скобки. Например: `TOPIC, (TOPIC, PART)`.

<*Идентификаторы итемов*> - или „пусто", или одиночное значение, или список значений, разделенных запятыми и заключенный в скобки. Значением здесь может быть либо литеральная строка, либо название атрибута контекста. Если правая часть условия опущена (пусто), то ее значением принимается ITEM_ID, т.е. идентификатор текущего итема.

Перед списком значений может быть указан так называемый *квантор*. Квантор — это одна из строк `any` или `all`. Назначение кванторов описано ниже.

<*Операция*> указывает на „критерий достижимости" искомых итемов из итемов заданных в правой части условия.

`<-` отбирает ближайших *потомков* заданных итемов по указанным типам связей.

`->` операция обратная `<-`. Отбирает ближайших *предков* заданных итемов по указанным типам связей.

`<--` отбирает итемы-потомки, достижимые из указанных в правой части по заданным типам связей.

`-->` отбирает итемы-предки, из которых достижим хоть один из заданных итемов по указанным типам связей.

Для условий связи с атрибутами контекста в правой части действует правило, похожее на используемое в условиях отношения: если в правой части условия связи указаны только атрибуты контекста, и все они не определены, то все условие „редуцируется", и не оказывает влияния на работу унифильтра.

Квантор `any` является умолчательным, т.е. его можно не указывать. Он означает, что отбираются итемы, связанные указанным образом *с любым* из итемов, перечисленных в списке. Квантор `all` означает, что отбираются итемы, каждый из которых связан *со всеми* итемами из списка.

Примеры унифильтров с условиями связи.

`{TOPIC <- 'default'}`

Отбирает итемы, ближайшие соседи итема 'default' по связи TOPIC.

`{TOPIC <- ITEM_ID}`
: То же, но для „текущего” итема, идентификатор которого находится в атрибуте контекста ITEM_ID.

`{TOPIC <-}`
: В точности то же, что и предыдущий пример. Пустая правая часть неявно заменяется на ITEM_ID.

`{TOPIC <--}`
: Все, а не только ближайшие потомки текущего итема по связи TOPIC.

`{(TOPIC, PART) <-- ('default', ITEM_ID)}`
: Все потомки текущего итема и итема 'default' по связям типов TOPIC и PART.

`{(TOPIC, PART) <-- all('default', ITEM_ID)}`
: Итемы, *каждый* из которых имеет предком по связям типов TOPIC и PART и текущий итем и итем 'default'.

`{AUTHOR -> ('default', ITEM_ID)}`
: Все авторы текущего итема и итема 'default'.

`{AUTHOR -> all('default', ITEM_ID)}`
: Авторы, которые *одновременно* являются авторами текущего итема и итема 'default'.

3.1.4 Предопределенные условия

Кроме описанных выше условий отношения и связи в тексте унифильтра возможно также употребление так называемых *предопределенных условий*.

sameserver

Условие `sameserver` отбирает итемы принадлежащие тому же серверу, что и текущий. То же условие может быть (более длинно) записано, как

```
{SERVER = CURRENT_SERVER}
```

Для условия `sameserver` используется следующее умолчание: оно неявно включается в фильтр, если этот фильтр не содержит ни одной операции связи. Т.е. при использовании только условий отношения все отобранные итемы будут принадлежать к текущему серверу.

Права доступа — allowed-only и include-forbidden

Эти два условия служат для ограничения (или наоборот — расширения) набора итемов, отбираемых фильтром, в зависимости от прав доступа текущего пользователя.

Условие `include-forbidden` отменяет проверку прав вообще. Т.е. при его указании фильтр будет отбирать даже итемы, которые обычно недоступны по чтению текущему пользователю. Условие `allowed-only` напротив, „включает" проверку прав доступа. Нельзя одновременно указывать оба этих условия.

Если фильтр не содержит ни одного из этих условий, то выполнение или нет проверки прав производится в соответствии со следующими правилами.

1. Права *не* проверяются, если текущий пользователь — не аноним, и имеет статус SUPERUSER.

2. Права *не* проверяются, если среди условий фильтра есть условие

   `BELONGS <linkop> ITEM_ID`

   где `<linkop>` — любое условие связи.

3. Права *не* проверяются, если среди условий фильтра есть

   `<LINKTYPE> <linkop> AUTHOR_ID`

   где `LINKTYPE` — название типа авторизующей связи а `linkop` — любое условие связи.

4. Права *не* проверяются, если сервер, к которому принадлежит текущий итем не требует авторизации.

5. Иначе — права проверяются, даже при отсутствии условия `allowed-only`.

3.2 Использование унифильтров

Унифильтры используются, в общем, так же, как и все остальные фильтры в Communiware - в различных динамических элементах, в первую очередь List и Loop. Так же, как и для SQL-фильтров, для унифильтров может быть указан список дополнительно извлекаемых атрибутов итемов и необходимый порядок их сортировки. При использовании операций `-->` и `<--` становится доступными атрибут DISTANCE, при использовании операции `<--` — атрибуты PARENT_ID и ORDNUM, а при использовании операций `->` и `<-` — атрибут ORDNUM.

3.2.1 Литеральные унифильтры

Как говорилось выше унифильтры, в отличие от SQL-фильтров, могут не храниться в БД как отдельные сущности, а записываться непосредственно в тексте шаблонов. Такие унифильтры будем называть литеральными. Вот пример работоспособного шаблона, использующего литеральные унифильтры.

```
<:Include *page_top:>
<:Loop "{TOPIC <-}" "+ORDNUM,TITLE,ABSTRACT":>
  <:Attr ORDNUM:>
  <:ItemLink ITEM_ID @TITLE:>
  <:Attr ABSTRACT:><br>
  <br>
<:EndLoop:>
<:Include *page_bottom:>
```

При просмотре итема по этому шаблону мы увидим список его непосредственных потомков итема по связи TOPIC, упорядоченных по ORDNUM.

3.2.2 Хранимые унифильтры

Так же, как и SQL-фильтры, тексты унифильтров могут храниться в базе данных. Такин унифильтры будем называть *хранимыми*. В этой секции описаны особенности использования хранимых унифильтров.

Создание и редактирование хранимых унифильтров

Создание и редактирование хранимых унифильтров (так же, как и SQL-фильтров) выполняется с помощью скрипта `filter`. Однако при работе с унифильтрами из этого скрипта надо отметить следующие особенности.

1. При создании нового хранимого унифильтра вместо текста фильтра на языке SQL в поле *Запрос* вводится текст на языке описания унифильтров, включая объемлющие фигурные скобки.

2. Значение полей *Разрешить переписывание* и *Значение по умолчанию* не играют роли.

3. В поле *Значения для тестирования* указываются имена и значения всех атрибутов контекста, используемых фильтром в следующем синтаксисе: `NAME1 => 'stringvalue', NAME2 => 123`. См. также ниже о параметризации унифильтров - значения параметров для тестирования передаются как значения атрибутов контекста `ARG1`, `ARG2` и т.д.

4. После нажатия кнопок *Тест* или *План выполнения* выводится также результат трансляции исходного текста унифильтра в SQL.

5. После первого выполнения тестирования или получения плана выполнения на странице скрипта появляется кнопка *Преобразовать в SQL*. При нажатии этой кнопки в поле *Текст* будет записан оттранслированный в SQL текст унифилтра, т.е. унифильтр станет „простым” SQL-фильтром. Следует отметить, что полученный SQL-текст и параметры фильтра могут потребовать „ручной” доводки.

Параметризация хранимых унифильтров

Хранимые унифильтры могут иметь набор позиционных параметров, значения которых указываются в шаблоне, в месте вызова унифильтра. Набор параметров указывается в поле *Аргументы* при работе со скриптом `filter`. Имена параметров разделяются запятыми. Однако имеет значение только количество аргументов, но не их названия. Таким образом указание в поле *Аргументы* `VAL1,VAL2` или `PUBLISED,TITLE`

полностью равноценны. Последним значением в списке аргументов может быть указана звездочка - *. Смысл ее, как и для SQL-фильтров, „все остальные параметры”.

При использовании хранимого унифильтра в шаблоне значения параметров указываются после имени фильтра в скобках — так же, как и для SQL-фильтров. При выполнении хранимого унифильтра значения параметров подставляются в атрибуты контекста с именами `ARG1`, `ARG2`. Значения параметров соответствующих звездочке записываются в атрибут контекста `ARGREST` как строка из значений, разделенных запятыми.

Предположим, что мы хотим создать хранимый унифильтр, отбирающий итемы заданного типа, имеющие дату публикации больше, чем указанная. Тогда при создании и тестировании унифильтра мы можем задать следующие значения:

```
Идентификатор:
  uf_example1
Запрос:
  {sameserver, TYPE_ID = ARG1, PUBLISHED > ARG2}
Аргументы:
  TID,PUB
Значения для тестирования:
  ARG1 => 'TOPIC', ARG2 => '2001.04.01 00.00.00'
```

Так как этот фильтр не использует идентификатор текущего итема, то `ITEM_ID` среди значений для тестирования можно не указывать.

Описанный метод параметризации хранимых унифильтров подходит для случая, когда параметризуются значения в правой части условий отношения, т.е. атрибуты контекста. Однако часто нужно параметризовать и другие части условий. Например типы связей в условиях связи или даже названия атрибутов *итема* в условиях отношения. В этих случаях применяется другой способ параметризации — текстовая подстановка. Дело в том, что в тексте хранимого унифильтра перед выполнением производится такая же подстановка значений атрибутов контекста на место конструкции `@ATTRNAME`, что и в динамическом элементе `Subst`. Поэтому для параметризации „не-атрибутов” в тексте хранимого унифильтра нужно использовать `@ARG1`, `@ARG2`, `@ARGREST` и т.п. Обращаем внимание на то, что при этом выполняется *текстовая* подстановка, т.е. апострофы, там, где они нужны в тексте фильтра, надо указывать самим.

Предположим, что мы хотим создать унифильтр, отбирающий потомков текущего итема по (нескольким!) указанным типам связей. Тогда поля скрипта `filter` можно заполнить следующим образом:

```
Идентификатор:
  uf_example2
Запрос:
  {sameserver, (@ARGREST) <-}
Аргументы:
  *
Значения для тестирования:
  ITEM_ID => 'default', ARGREST => 'TOPIC,PART'
```

Теперь этот фильтр может быть использован где-нибудь шаблоне примерно так: `<:Loop "uf_example2(TOPIC,PART)TITLE":>...`

Уточнение условий в хранимых унифильтрах

Использование хранимых унифильтров в шаблонах позволяет выразить следующее: „хочу получить из итемов, отбираемых таким-то хранимым унифильтром удовлетворяющие следующему условию . . . ”. Т.е. возможно добавить к условиям унифильтра еще какие-то условия, задав их непосредственно в тексте шаблона.

Для этого в тексте шаблона (в динамических элементах `Loop` и `List`) после имени хранимого унифильтра нужно указать вертикальную черту (|), а затем, в фигурных скобках, дополнительные условия на языке описания унифильтров. Например конструкция

```
<:Loop "uf_example2(TOPIC,PART)
                  | {PUBLISHED > SYSDATE - 10}":>
```

возвратит список итемов — непосредственных потомков текущего по связям TOPIC и PART (результат работы `uf_example2`), но только те из них, которые опубликованы за последние десять дней.

Глава 4

Структура базы данных Communiware

Разработчику SQL-фильтров приходится непосредственно иметь дело с базой данный Communiware. Для того, чтобы правильно сформулировать SQL-запрос, необходимо знать как распределяются данные по таблицам, и какие отношения между этими таблицами можно установить.

4.1 Таблица ITEM

Основной таблицей Communiware является таблица ITEM. Ключом этой таблицы является поле ITEM_ID, содержащее уникальный идентификатор итема, который используется как составная часть ключа практически во всех остальных таблицах Communiware, с которыми придется иметь дело разработчику фильтров.

Кроме ITEM_ID таблица ITEM содержит ряд атрибутов, которыми обладают все итемы.

TITLE Заголовок итема. Имеет тип RICHTEXT, т.е. может содержать HTML-ную разметку. Достаточно часто извлекается неперекрываемыми фильтрами, но практически никогда не используется при сравнениях. Иногда, в этом поле производится поиск с использованием оператора LIKE, но его следует избегать, т.к. поиск по

LIKE требует полного сканирования таблицы, а это одна из самых больших таблиц в базе.

WRITTEN Тип DATE. Дата написания итема (в реальном мире).

PUBLISHED Тип DATE. Дата публикации итема на сайте. Очень часто используется и для поиска, и для сортировки.

LASTCHANGE Тип DATE. Дата последнего изменения итема. Это поле автоматически изменяется триггерами при любой модификации таблицы ITEM или таблицы расширенных атрибутов, если она у итема есть, и даже иногда при модификации таблицы ITEM_LINK. Этот атрибут используется для вычисления даты модификации страницы, на которой встречается данный итем (т.е. используется всегда), но его использование в содержательных целях как правило бессмыслено.

ABSTRACT Тип RICHTEXT. Аннотация (краткое содержание) итема. Достаточно часто визуализируется в шаблонах, но в условиях выборки используется еще реже, чем TITLE.

TYPE_ID Тип STRING. Является внешним ключом, ссылающимся на таблицу ITEM_TYPE (раздел 4.5.1). Идентификатор типа итема. Очень часто используется в качестве критерия отбора, но практически никогда — в качестве критерия сортировки, так как лексикографическая упорядоченность типов обычно содержательного смысла не имеет.

Проверка на попадание TYPE_ID в определенный список — один из наиболее частых случаев использования множественного аргумента (см. раздел 2.2.2).

TEMPLATE_ID Тип STRING. Идентификатор шаблона по умолчанию. Используется в фильтрах крайне редко.

PARAMS Тип STRING. Атрибут, зарезервированный для использования разработчиками конкретных онтологий и сайтов. Т.е. разработчик онтологии может использовать его для какой-нибудь специфической цели в своем типе итема.

TEXTSIZE Тип NUMBER. Размер текста итема в символах без учета HTML-разметки. Может быть использован например для подсчета статистики по авторам.

DATASIZE Тип NUMBER. Физический размер текста итема, хранящегося в базе. Имеет смысл использовать для проверки на наличие у итема текста, так как сам текст хранится в поле типа LOB, которое нельзя использовать в предложении WHERE SQL-запроса.

SERVER Тип STRING. Идентификатор сайта, к которому принадлежит итем. Очень часто используется при отборе итемов. Следует учесть что атрибут контекста SERVER содержит идентификатор сайта, которому принадлежит итем, создавший данный контекст, а не идентификатор сайта, через который мы смотрим на текущую страницу (эти два сайта могут не совпадать. Например, автор был зарегистрирован на соседнем сайте, а нас интересует список его работ на данном). Поэтому сравнивать это поле имеет смысл с атрибутом контекста CURRENT_SERVER.

OLD_URL Тип STRING. Исходно задуман для хранения старой URL страницы статического сайта, импортированной в Communiware. Реально применяется для хранения любых значений с семантикой URL, например для URL внешней ссылки или для имени файла, являющегося основным содержимым итема с Content-Type=Binary.

STATUS Тип STRING. Содержит набор значенй, специфичных для данного типа итема, определяющих его состояние. Является компонентом внешнего ключа, ссылающегося на таблицу STATUS, ключ которой (TYPE_ID,STATUS). Значения статуса упорядочены, но не лексикографически, а по значениям поля ORDNUM таблицы STATUS. Поэтому, если необходимо использовать сравнение статусов по больше/меньше, используется соединение с таблицей STATUS и некореллированный подзапрос для получения ORDNUM того статуса, с которым сравнивать.

Например.

```
select item_id from item,status where
item.type_id=? and item.type_id=status.type_id
and item.status=status.status and
status.ordnum>(select ordnum from status where
type_id=? and status=?)
```

Аргументами такого фильтра, очевидно, должны быть TYPE_ID, TYPE_ID, STATUS.

WRITTEN_ACCURACY Тип STRING. Внешний ключ DATE_ACCURACY Точность представления даты написания. Используется в основном для форматирования дат при выдаче.

STATUS_UPDATE Тип DATE. Обновляется триггером в момент изменения поля STATUS. Очень часто используется для отбора и сортировки в Workflow-приложениях.

RESTRICT_TEMPLATES Тип NUMBER. Используется ядром Communiware при проверке допустимости показа данного итема в данном контексте.

KEYWORDS Тип STRING. Список ключевых слов через запятую. При поиске лучше использовать таблицу KEYWORDS, где данное значение разбито на ключевые слова. С появлением полноценных множественных атрибутов будет удалено.

К набору стандартных атрибутов итема примыкает также таблица KEYWORDS, содержащая поля ITEM_ID и KEYWORD. В этой таблице может быть несколько записей на один итем. В поле KEYWORD каждая запись содержит одно ключевое слово из списка введенных через запятую в поле KEYWORDS таблицы ITEM. Поддерживается постпроцессором Keywords, который необходимо вызывать из формы постинга явным образом. Это крайне рекомендуется делать, если необходим поиск по ключевым словам.

Ключевые слова в таблице KEYWORDS приведены к нижнему регистру.

4.2 Таблицы расширенных атрибутов

Некоторые типы итемов Communiware кроме стандартных атрибутов имеют так называемые расширенные.

Каждому такому итему соответствуют записи в двух таблицах Communiware — таблице ITEM и таблице расширенных атрибутов, соответствующей данному типу. Как правило, каждый тип имеет свою собственную таблицу расширенных атрибутов, имя которой совпадает с именем типа, но возможны ситуации, когда несколько типов разделяют общую таблицу расширеных атрибутов.

Ключом таблицы расширенных атрибутов является ITEM_ID. Использовать таблицы расширенных атрибутов в запросе обычно приходится только в тех случаях, когда на них накладываются условия. Если нужно просто извлечь расширенные атрибуты, с этим обычно справляется либо машина переписывания, либо механизм контекстов.

Все поля таблицы расширенных атрибутов должны быть описаны в таблице ATTRIBUTE как атрибуты классов UPDATABLE, COMPUTED или INTERNAL. Просмотреть их можно через позицию «Типы итемов» в меню интерфейса вебмастера.

Следует учесть, что если вы выбираете фильтром итемы различных типов и используете естественное соединение с таблицей расширенных атрибутов, из результата запроса будут выброшены все итемы, у которых нет расширенных атрибутов, или используется другая таблица расширенных атрибутов. В этих случаях надо использовать внешнее соединение.

4.3 Таблицы ITEM_REL и ITEM_LINK

Как правило, при конструировании фильтров приходится иметь дело не только, и не столько с атрибутами итема, сколько со связями, которые он имеет с другими итемами.

Информация о связях хранится в таблицах ITEM_LINK и ITEM_REL.

Первая из них содержит информацию о связях, непосредственно связывающих два итема, т.е. дуги графа, образованного итемами и связями. Вторая — содержит информацию о путях в этом графе длиной более 1, которые разработчиком онтологии сочтены интересными. Это либо пути, проходящие по одинаковым связям, например путь из рубрики в подрубрику ее подрубрики, либо пути проходящие по цепочке связей, объекдиненных в метасвязь, например путь из рубрики в реплику в дискуссии по главе книги, входящей в подрубрику этой рубрики.

Соответственно, в запросах обе таблицы используются примерно с одинаковой частотой, в зависимости от того, нужны ли нам только ближайшие предки или потомки, или вся «родословная».

4.3.1 Структура таблицы ITEM_LINK

ACTIVE Идентификатор итема, который является в данной связи активным (предок, который породил — активный залог).

PASSIVE Идентификатор итема, который является в данной связи пассивным (потомок, который порожден — пассивный залог)

LINKTYPE_ID Идентификатор типа связи. Внешний ключ на таблицу LINKTYPE.

ORDNUM Тип NUMBER. Порядковый номер данной связи среди всех связей одного типа, ведущих к одному предку (активному итему). Используется для сортировки потомков, когда не задано другого критерия сортировки.

Ключом в таблице ITEM_LINK является комбинация (ACTIVE, PASSIVE, LINKTYPE_ID).

4.3.2 Структура таблицы ITEM_REL

ACTIVE Идентификатор начала пути,

PASSIVE Идентификатор конца пути,

LINKTYPE_ID Идентификатор типа (мета)связи.

DISTANCE Длина минимального пути данного типа, соединяющего два данных итема.

REFCOUNT Внутренний счетчик, используемый триггерами, обновляющими ITEM_REL при изменениях ITEM_LINK. Вообще-то равен числу путей данного типа, соединяющих два итема, но вы уверены, что триггер считает пути так же как вы?

4.4 Итемы с табличным содержимым

В некоторых случаях в Communiware хранятся данные, ради каждого значения которых не имеет смысла заводить отдельный итем. Например, временные ряды курсов валют, метеорологические данные, реплики в чате. Все эти данные являются частями некоторой большой сущности (курса данной валюты, погоды в данной точки, chatroom), имеющей внутри явно списочную или табличную природу. С целью удобства обработки (например, построения графиков) такие значения осмыслено хранить в виде отдельных записей в таблице для каждого значения,

или для кортежа значений, относящихся к одному моменту времени, одной точке в пространстве и т.д.
Такие данные представляются в Communiware в виде итемов с Content-Type=TABLE.

Каждому типу итема с таким типом содержимого обычно соответствует отдельная таблица, ключ которой включает ITEM_ID, но содержит еще и какое-то другое поле (например временную отметку) или поля, позволяющие различить записи внутри одного итема.

И ключевые, и информационные поля такой таблицы описаны в таблице ATTRIBUTE как атрибуты соответствующего класса (TABLE_KEY и TABLE_VALUE). Описание их можно просмотреть через позицию «Типы итем» в меню интерфейса вебмастера, или найти в описании конкретной онтологии.

4.5 Описание типов итемов, связей и атрибутов

Таблицы, описывающие метасущности — типы итемов, типы связей и атрибуты, крайне редко приходится использовать при написании фильтров. Тем не менее, опишем и их для того чтобы наше описание структуры базы данных Communiware могло претендовать на полноту.

4.5.1 Таблица ITEM_TYPE

Описывает типы итемов.

TYPE_ID идентификатор типа. Первичный ключ.

TEMPLATE_ID идентификатор шаблона по умолчанию (глобального, используемого если не задан шаблон на уровне сайта). Обычно это шаблон с сайта default.

PARAM Зарезервированное поле. Не используется.

DC_CODE На ранних этапах развития Communiware предполагалось обеспечить совместимость со стандартном библиографической информации Doublin Core. Communiware давно переросла рамки системы публикации и далеко не все типы итем допускают описание средствами библиографии, но поле для кода по Doublin Core в описании типа осталось.

EXTENDED_ATTR_TABLE Имя таблицы расширеных атрибутов

MSG_ID Идентификатор текстового описания итема. О системе интернационализированных текстовых описаний более подробно рассказано в разделе 4.6.

CONTENT_TYPE Тип содержимого итема, вернее имя модуля из иерархии `Communiware::Datatype` реализующего операции с данным типом значений. Может быть Html, Template, Binary, None, Code или Table.

DATA_TABLE Для итемов с CONTENT_TYPE='Table' — имя таблицы с данными итема. Для всех прочих — NULL.

4.5.2 Таблица LINKTYPE

Описывает типы связей. Содержит гораздо больше полей, которые принципиально могут интересовать автора фильтров, поскольку позволяют группировать связи в содержательные группы.

LINKTYPE_ID Идентификатор типа связи. Первичный ключ.

HIERARCHY Число 0 или 1. Если 1, то данная связь иерархическая, т.е. пути по цепочке таких связей записываются в таблицу ITEM_REL.

BIDIRECTIONAL Число 0 или 1. Если 1, то связь двунаправленная, т.е. активный и пассивный концы эквивалентны. Реально при создании двунаправленной связи в таблицу ITEM_LINK вносятся две записи, отличающиеся порядком следования активного и пассивного итема.

REL_TYPE Тип отношения. Принимает значения 1:1, 1:N и *M*:N. 1:N означает, что у данного активного итема может быть сколько угодно пассивных соседей по этой связи, но у данного пассивного — только один активный.

EDITABLE Число 0 или 1. Значение 0 в этом поле означает, что эта связь специальным образом поддерживается кодом ядра Communiware. 1 — обычная связь, которую можно редактировать через разнообразные интерфейсы Communiware.

META Число 0 или 1. Значение 1 означает, что это метасвязь, т.е. в таблице ITEM_LINK не должно быть экземпляров данной связи, только пути по ней могут появляться в ITEM_REL.

PASSABLE Число 0 или 1. Принимает значение 1 либо если HIERARCHY=1, либо если связь входит в метасвязь. При создании или удалении связи с PASSABLE=1, триггерам Communiware вероятно, потребуется произвести какие-то операции с ITEM_REL.

AUTHORIZE Число 0 или 1. Если связь авторизующая, то наличие ее у персоны с каким-либо итемом, дает этой персоне как минимум права на чтение этого итема, а возможно еще какие-либо дополнительные.

MSG_ID Идентификатор текстового описания типа связи. См.раздел 4.6

Аналогично тому, как над таблицей ITEM надстроены таблицы ITEM_LINK и ITEM_REL, над таблицей LINKTYPE надстроены таблицы META_LINK и METALINKS_REL, реализующие группировку типов связей в иерархию метасвязей. Обращаться к ним при создании фильтров не требуется никогда, так как для поиска дуг в графе, входящих в метасвязь проще выбрать из ITEM_REL записи с LINKTYPE_ID соответствующей данной метасвязи и DISTANCE=1.

4.5.3 Таблица ALLOWED_LINKTYPE

Далеко не все типы итемов могут участвовать во всех типах связи. Более того, некоторые типы итем могут быть активными в данном типе связи, но не пассивными или наоборот. Например, статья может входить в рубрику (быть пассивной в связи TOPIC) но не может быть рубрикой для какого-то другого итема (быть активной в этой связи). С другой стороны статья может быть предметом дискуссии (быть активной в связи PARENT) но не может быть репликой в этой дискуссии.
Подобные ограничения описываются таблцей ALLOWED_LINKTYPE.

Ее структура следующая:

TYPE_ID Тип итем

LINKTYPE_ID Тип связи

LINKSIDE принимает значения ACTIVE и PASSIVE.

Ключевыми здесь являются все три поля.

4.5.4 Таблица ATTRIBUTE

Атрибуты в Communiware это не просто колонки в таблице. С атрибутами, во всяком случае с теми из них, которые являются неотъемлемыми свойствами итемов определенных типов связано довольно много информации — в какой таблице хранится, какой имеет тип, может ли изменяться пользователями, какие значеня может принимать.

Эта информация хранится в таблице ATTRIBUTE.

Ее структура:

NAME Имя атрибута

ATTR_TYPE Тип атрибута. Один из STRING, NUMBER, DATE, RICHTEXT.

ATTR_STATE Класс атрибута. Может быть редактируемым (UPDATABLE), вычислимым (COMPUTED), внутренним (INTERNAL), настройкой пользователя (COOKIE), групповым (QUALIFIED), ключевым полем таблицы значений (TABLE_KEY) и полем данных в ней (TABLE_VALUE).

TABLE_NAME Имя таблицы, в которой хранится атрибут. Не имеет смысла для QUALIFIED, COOKIE и части COMPUTED атрибутов.

QUERY текст SQL-запроса для получения значения атрибута по ITEM_ID. Основное свойство для группового атрибута или для вычислимого, для которого не определена таблица хранения. Если задана таблица, то, вероятно запрос был сконструирован автоматически — извлечь все поля из этой таблице, которые упоминаются в таблице ATTRIBUTE.

LOOKUP_TABLE Имя таблицы значений данного атрибута. Если определена, то атрибут может принимать только значения присутствующие в поле LOOKUP_COLUMN этой таблицы. (При создании таблиц значений через Web-интерфейс или интерфейс загрузки онтологии Communiware автоматически создает и соответствующий integrity constraint).

LOOKUP_COLUMN Имя ключевого поля таблицы значений.

MENU_COLUMN Имя поля, значения которого необходимо выводить при формировании меню значений данного атрибута. Конструкция TXT(атрибут) в шаблоне Communiware выводит именно текст, ассоциированный с текущим значением атрибута посредством MENU_COLUMN.

LOOKUP_KEY Дополнительное ключевое поле таблицы значений. Например, для атрибута STATUS LOOKUP_KEY равен TYPE_ID.

DEFAULTS Значение атрибута по умолчанию. Имеет смысл только для атрибутов класса COOKIE.

MSG_ID Идентификатор краткого текстового названия атрибута

MSG_LONG Идентификатор длинного поясняющего текста.

4.6 Таблицы значений атрибутов

Некоторые атрибуты в Communiware могут принимать только ограниченное множество значений. Для таких атрибутов создаются таблицы значений.

Вообще говоря, таблицей значений следует считать любую таблицу, связанную с полем атрибута ограничением ссылочной целостности. В этом смысле таблицей значения для поля TYPE_ID является таблица ITEM_TYPE, а для поля TEMPLATE_ID — таблица TEMPLATE, которая одновременно является таблицей расширеных атрибутов типа итем TEMPLATE. (то же касается атрибута SERVER и таблицы SERVER).

Мы рассмотрим в этом разделе только те таблицы значений, которые не несут никакой другой функции, кроме задания списка значений атрибута.

Как правило, такие таблицы состоят из двух полей — ключевого поля, по имени и типу совпадающего с полем самого атрибута, и поля расшифровки, используемого при генерации меню. Это может быть либо строковое поле с текстовой расширфорвкой, которая будет единой для всех языков, либо числовое поле с именем MSG_ID (в таблице ATTRIBUTE есть еще MSG_LONG) с аналогичной семантикой.

Число хранимое в этом поле является входом в таблицу интернационализированных сообщений Communiware. (называется MESSAGE и имеет три поля MSG_ID, LANG и TEXT).

Доступ к тексту сообщения осуществляется с помощью хранимой функции `get_msg`, которая выбирает текст сообщения на том языке, котоорый был выбран в начале сессии.

Если в запросе употребить конструкцию `SELECT get_msg(msg_id) as "TXT(TYPE_ID)"` результатом будет текстовое описание на текущем языке.

Кроме ключевого поля и поля текстового описания, в таблицах значений может быть еще поле ORDNUM позволяющее задать упорядоченность на для строковых значений атрибутов. Впрочем, упорядочивание по полю ORDNUM таблицы значений машиной переписывания фильтров пока не поддерживается.

Кроме того, в Communiware предусмотрен механизм, позволяющий переключать наборы допустимых значений в зависимости от некоторого другого атрибута итема, например TYPE_ID. Реально он используется в настоящее время только для атрибута статус.

4.7 Таблицы подписки

Механизм подписки в Communiware напоминает по своей структуре связи, но это связи не между двумя, а между тремя итемами — адресатом подписки, итемом, на обновления в котором подписываются и шаблоном, по которому генерируется дайджест обновлений. Понятие «обновление в итеме» в данном случае следует понимать расширено. Например, «обновление в рубрике» это появление в этой рубрике новых статей.

Вообще, «обновление в итеме» с точки зрения подписки, это любые изменения в базе данных Communiware, которые можно отследить с помощью фильтра, работающего в контексте данного итема.

Информация о подписке хранича в таблице SUBSCRIPTION, которая имеет следующую структуру:

ITEM_ID итем, изменения в котором посылаются по подписке

ADDRESSEE пользователь, которому послылается данная подписка

SUBSCRIPTON_TYPE шаблон, по которому форматируются письма. В нем же задается фильтр, с помощью которого определяется необходимость отправки письма.

PERIOD Период проверки на наличие обновлений. Внешний ключ на таблицу SUBSCRIPTION_PERIOD.

FORMAT_ID Формат отправки писем. Может быть либо text/html, либо text/plain.

Таблица SUBSCRIPTION_PERIOD имеет следующую структуру

PERIOD Идентификатор периода, тот же что используется в таблице SUBSCRIPTION и указывается в командной строке cmwmaild при запуске из cron.

MSG_ID Идентификатор текстового описания периода (см. раздел 4.6

LENGTH Число, продолжительность периода в секундах.

LAST_SENT Дата, время последней обработки подписки с данным периодом. При проверке на обновления отбираются все итемы, у которых проверяемая дата (зависит от фильтра подписки, см раздел 1.2.3) находится в интервале от LAST_SENT до текущй даты.

4.8 Вспомогательные таблицы для работы с шаблонами

Шаблоны Communiware являются итемами, и поэтому значительная часть их взаимоотношений между собой и с другими итемами описывается посредством связей. Наличие поля TEMPLATE_ID в таблице ITEM — всего лишь оптимизация производительности (так называемая денормализация).

Тем не менее, шаблоны находятся в некоторых отношениях с другими объектами, которые итемами не являются, а именно с типами итемов и фильтрами.

Таблица TEMPLATE_FILTER перечисляет фильтры используемые в данном шаблоне. Она заполняется автоматически при сохранении шаблона, когда его текст анализируется с целью поиска синтаксических ошибок. Ее основное назначение — облегчение навигации по шаблонам — она позволяет вывести на страницах просмотра и редактирования шаблона список используемых фильтров, и запрещение удаления фильтра, который используется в шаблонах.

Таблица SUBS_ITEM_TYPE описывает отношение шаблонов и типов итемов. Исходное ее назначение — указание того, какие шаблоны подписки можно использовать для подписки на итемы определенного типа. Впоследствии она стала также использоваться и для того, чтобы указать что шаблон страницы может быть шаблоном по умолчанию для заданного типа. Она заполняется автором шаблона при редактировании его с помощью шаблона edit_template.

4.9 Шаблоны по умолчанию

В Communiware используется достаточно сложный механизм назначения шаблонов по умолчанию. Если при создании итема его шаблон не указан явно, производится поиск вверх по дереву рубрик, и если у одного из итем в этом дереве обнаружился список шаблонов по умолчанию, содержащий шаблон для данного типа, то используется он. Иначе, проверяется список шаблонов по умолчанию для текущего сайта, и если не обнаружился подходящий шаблон там, используется глобальный шаблон по умолчанию, идентификатор которого хранится в таблице ITEM_TYPE.

Таблица шаблонов по умолчанию уровня сайта играет еще и роль ограничителя списка типов доступных на сайте. Стандартный интерфейс вебмастера не позволяет создать итем такого типа, для которого нет шаблона по умолчанию на уровне сайта.

Таблицы шаблонов по умолчанию, привязанные к сайтам и другим структурообразующим итемам, например рубрикам, хранятся в таблице DEFAULT_TEMPLATE, структура которой следующая:

KEYITEM Идентификатор ключевого итема, т.е. того, к которому относится данный вход в таблицу шаблонов.

TYPE_ID Тип итем, к которому будет применяться данный шаблон

TEMPLATE_ID Идентификатор шаблона.

Ключом здесь является пара (KEYITEM, TYPE_ID).

4.10 Виртуальные итемы

Механизм виртуальных итемов является некоторой компенсацией глобальности пространства ITEM_ID в Communiware.

Он позволяет ссылаться на некоторые итемы (обычно шаблоны) не по их имени, а по имени функции, которую они выполняют на сайте.

Этот механизм реализован с помощью двух таблиц — таблицы SPECPAGE_TYPE, описывающей возможные типы виртуальных итем, т.е. функции которые они выполняют, и таблицы SPECPAGES, перечисляющей конкретные итемы, выполняющие данные функции на конкретном сайте.

В соответствии с данным определением таблица SPECPAGES состоит из трех полей — SERVER, PAGETYPE и ITEM_ID, причем ключом является пара (SERVER, PAGETYPE).

Таблица SPECPAGE_TYPE задащая возможные PAGETYPE несколько сложнее.

PAGETYPE Идентификатор типа виртуальных итем, или обозначение функции, которую выполняет данный виртуальный итем.

MSG_ID Ссылка на интернационализированное название данной функции.

TYPE_ID Идентификатор типа итем, который может выполнять данную фунукцию

TEMPLATE_TYPE Если TYPE_ID=TEMPLATE, это поле содержит значение расширенного атрибута шаблона TEMPLATE_TYPE, соответствующего выполняемой функции.

Глава 5

Переписывание фильтра

Машина переписывания фильтров Communiware — это программный модуль, который анализирует и переформулирует текст SQL-запроса фильтра с целью извлечения из базы нужного набора атрибутов за один проход, а также с целью упорядочения результатов фильтра.

Благодаря наличию этой машины, одни и те же фильтры можно использовать в очень большом числе шаблонов.

К сожалению, машина переписывания имеет свои ограничения. Написание полноценного парсера SQL — сложная задача, решать которую в полном объеме не было необходимости. Написание генератора оптимальных SQL-запросов — тоже. Тем более, если бы эту задачу удалось решить, можно было бы использовать для фильтров Communiware синтаксис, более адекватно отражающий графовую модель Communiware, чем SQL.

В большинстве случаев машина переписывания в состоянии отдетектировать непереписываемый фильтр сама. Но иногда ее, в значительной степени эвристическая, логика может дать сбой и в результате переписывания получится некорректный SQL-запрос, или, что еще хуже, запрос, дающий неверный результат. Возможно также, что в результате переписывания существенно ухудшится время выполнения запроса.

Поэтому все вновь разрабатываемые фильтры следует тестировать на корректное переписывания (для этой цели нужно ввести какие-нибудь имена атрибутов в строку «Атрибуты для извлечения» в редакторе фильтров и нажать кнопку «Тест» или «План выполнения»)

В случае, если переписывание фильтра дает неадекватные результа-

ты, следует убрать отметку с чекбокса «Разрешить переписывание».

В этом случае фильтр будет всегда выполняться в том виде, в каком он был вами введен, а недостающие атрибуты, используемые в шаблонах, извлекаться дополнительными SQL-запросами.

Сортировка при этом будет производиться Communiware, а не сервером баз данных.

Если вы объявили фильтр непереписываемым, следует позаботиться о том, чтобы он извлекал наиболее часто используемые атрибуты даже в непереписанном виде. В частности, атрибут LASTCHANGE следует извлекать всегда, так как он используется интерпретатором шаблонов при любой обработке контекста итема.

Существенно важным моментом является то, что сортировка непереписываемого фильтра возможна только по тем атрибутам, которые он извлекает. Поэтому все атрибуты, которые могут быть использованы в качестве критериев сортировки также следует извлекать.

Поскольку непереписываемые фильтры обычно имеют весьма ограниченную область применения — в одном-двух шаблонах в рамках конкретного сайта, разработчик такого фильтра обычно представляет себе, какие атрибуты могут понадобиться.

5.1 Переписывание с целью извлечения атрибутов

5.2 Переписывание с целью упорядочения

5.3 Ограничения, накладываемые на переписываемые фильтры

Глава 6

Поддерживаемые диалекты SQL

Глава 7

Оптимизация производительности фильтров

7.1 Профилирование шаблонов

Выполнение фильтров — наиболее ресурсоемкая часть работы Communiware. К счастью, это одновременно и наиболее легко поддающаяся оптимизации часть. Поддерживаемые SQL-сервера, в особенности Oracle предоставляют обширный набор средств для настройки производительнотси SQL-запросов.

Но прежде чем заниматься оптимизацией производительности конкретного фильтра, необходимо убедиться, что проблема именно в нем. Для этой цели в Communiware предусмотрен режим профилирования. Он, как и другие отладочные режимы Communiware включается посредством указания в URL соответствующего параметра. Допишите в URL `?PROFILE=1` (или, если в URL уже есть параметры `&PROFILE=1`, и будет включен режим профилирования шаблонов. Текущее состояние отладочных параметров запоминается в куке, и будет действовать до тех пор, пока вы его не выключите явным образом, указав PROFILE=0.

При включенном режиме профилирования в конце каждой страницы показывается табличка, показывающая суммарные затраты времени на все операции с базой данных (см. рис. 7.1). Каждый фильтр показан в

Profiling info

ID	Parsing		Execution		Fetching		
	count	time	count	time	count	count	rows
AllOpen_MultiTyped	1	0.044176	1	2.246934	1	0.001083	5
get_qualified_attr	72	0.090646	72	0.153715			
selectrow_cached			53	0.047966			
checktemplate			8	0.028393			
getattr			11	0.022861			
AllOpenTyped	1	0.006216	1	0.012409	1	0.00031	
ActiveLinked	1	0.003925	5	0.00528	5	0.001762	
TypedTree	1	0.001295	1	0.002349	1	0.005857	18
PassiveLinkedVirtual	1	0.001051	1	0.001084	1	0.000315	
Total:	77	0.147309	153	2.520991	9	0.009327	23

Total execution time: 3.149075 sec
Additional attributes requested: ITEM_TYPE(5) LASTCHANGE(5) SYSDATE(1)

Рис. 7.1: Таблица профилирования шаблона

этой таблце отдельной строкой, причем они отсортированы в порядке убывания времени, затраченного на их обработку.

Поэтому легко определить, какой из фильтров является „узким местом” при обработке данного шаблона. Имена фильтров в этой табличке являются ссылками на интерфейс редактирования фильтров, что позволяет легко перейти от профилирования шаблона к оптимизации данного фильтра.

Время выполнения фильтра в таблице профилирования разбито на три части: parsing, execution и fetching. Parsing это время разбора SQL-запроса и формирования плана выполнения. Поскольку разобранные запросы кэшируются SQL-сервером, это время как правило пренебрежимо мало. Execution, это время собственно выполнения запроса. Обычно оптимизация направлена на уменьшение как раз этого времени. Fetching — время передачи значений из SQL-сервера в Communiware. Обычно оно прямо пропорционально количеству извлекаемых записей. Иногда возникают ситуации, когда время выполнения запроса относительно мало, а время извлечения данных недопустимо велико.

Происходит это в тех случаях когда реально запрос выполняется по мере извлечения данных, а как собственно время выполнения учитывается только выполнение некоторых подготовительных операций. Если

```
Здесь должен быть план}
```

Рис. 7.2: План выполнения SQL-фильтра

добавить к такому запросу предложение order by, то все временные затраты переместятся в раздел исполнения. Впрочем, если запрос возвращает несколько сотен строк, а используются только первые несколько из них, имеет смысл использовать именно такой режим.

7.2 Анализ плана выполнения

План выполнения SQL-запроса это реальная последовательность обращений к таблицам и индексам, выполняемая сервером в процессе выполнения запроса.

План выполнения формируется оптимизатором SQL-сервера на основе большого набора эвристических правил. Тем не менее, автоматически сформированный план не всегда оптимален. Например, на оптимальность плана влияет точность имеющихся сведений о статистике распределения значений в колонках таблицы. Эту статистику можно вычислить с помощью команды `ANALYZE` в Oracle и `VACUUM ANALYZE` в Postgresql.

Скрипт резервного копирования Communiware автоматически обновляет статистику по наиболее часто используемым таблицам Communiware.

Признаком неоптимальности плана выполнения запроса достаточно часто является наличие в нем операции полного сканирования большой таблицы. Например, появление в плане FULL TABLE SCAN (ITEM) (или, еще хуже, ITEM_REL) требует принятия срочных мер. С другой стороны, полное сканирование маленькой таблицы, например какой-нибудь таблицы значений состоящей из 5–7 строк, как правило быстрее любого другого способа доступа к ней, особенно если в опциях хранения этой таблице указано «кэшировать в памяти».

Способам оптимизации плана выполнения запросов посвящена значительная часть документации на SQL-сервер. Мы разберем здесь только наиболее общие методы оптимизации запроса.

7.3 Переформулировка SQL-запросов

Наиболее общим и мощным методом оптимизации является переформулировка запроса. Сложные критерии отбора записей как правило можно выразить несколькими способами, эквивалентными с точки зрения реляционной алгебры, но отнюдь не эквивалентными с точки зрения производительности конкретного SQL-сервера.

Например, соединение таблиц часто можно заменить кореллированным подзапросом и наоборот. Поскольку соединение таблиц более употребительная синтаксическая конструкция, зачастую замена проверки exists или not exists на соединение (даже внешнее) приводит к ускорению выполнения запроса, даже если приходится использовать distinct или group by для достижения эквивалентности запросов.

7.4 Создание индексов

Использование индексов в SQL базах данных практически никогда не влияет на набор записей, возвращаемых запросом, но достаточно часто очень сильно влияет на производительность запросов.

Если удается определить, какому условию в запросе соответствуют операции полного сканирования таблицы или сканирования диапазона по плохо селективому индексу, создание подходящего индекса часто позволяет ускорить запрос в десятки раз. Зачастую, просто создание индекса по тем же колонкам, но в другом порядке, позволяет заменить FULL SCAN на существено более эффективный RANGE SCAN, а то и UNIQUE SCAN.

Communiware в настоящее время не предоставляет web-интерфейсов для создания индексов[1]. Поэтому для создания индекса вам придется воспользоваться родными средствами используемого SQL-сервера.

7.5 Использование хинтов

Oracle предоставляет разработчику средства управления планом выполнения запроса, называемые хинтами. Это комментарии специального формата, позволяющие рекомендовать оптимизатору использовать или не использовать определенный индекс, производить доступ к указанной

[1] в TODO

таблице путем полного сканирования и так далее. Синтаксис хинтов описывается в книге «Oracle 8 SQL Reference». Следует учитывать что хинты носят рекомендательный характер, и, в принципе, могут быть проигнорированы SQL-сервером. В PostgreSQL аналогичного механизма пока нет.

Глава 8

Групповые атрибуты — ближайшие родственники фильтров

Приложение А

Список стандартных фильтров