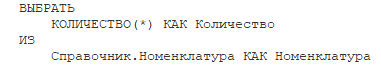
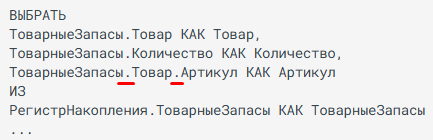
**Запросы**

**1.** Чтобы получить количество элементов в таблицы можно использовать такой запрос:



**2.** Разыменование полей. Использование более N точек в запросе может привести к понижению производительности.

В языке запросов возможно обращаться не только к полям исходных таблиц запроса, перечисленных в предложении ИЗ, но и к полям таблицы, на которую ссылается поле исходной таблицы запроса, если это поле имеет ссылочный тип. Имена полей при этом пишутся "через точку". Применение такой конструкции приводит к неявному соединению с дополнительными таблицами для получения значений полей "через точку".  
Например:

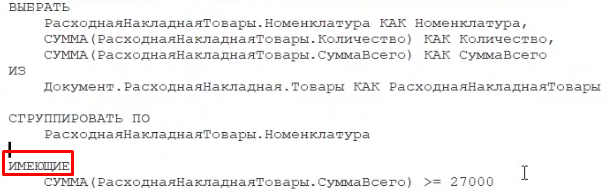
Большое число исходных таблиц запроса приводит к его усложнению и может значительно увеличивать время его выполнения.

Особенно это важно помнить в тех случаях, когда поле таблицы ссылочного типа имеет составной тип и может содержать ссылки на несколько таблиц. В таком случае, получение полей других таблиц "через точку" от такого поля составного типа приведет к соединению со всеми таблицами, ссылки на которые могут оказаться в данном поле и в RLS к этим таблицам.

Подобное получение данных "через точку" от **ссылочных полей составного** типа крайне нежелательно. Каждое исключение из этого правила должно тщательно анализироваться.

* Как правило, для выполнения конкретного запроса в данных условиях не нужны все возможные типы данной ссылки. В этом случае, следует ограничить количество возможных типов при помощи функции ВЫРАЗИТЬ.
* Если данный запрос является универсальным и используется в нескольких разных ситуациях (где типы ссылки могут быть разными), то можно формировать запрос динамически, подставляя в функцию ВЫРАЗИТЬ тот тип, который необходим при данных условиях. Это увеличит объем исходного кода и, возможно, сделает его менее универсальным, но может существенно повысить производительность и стабильность работы запроса.
* Разыменование это не плохо, если оно делается над полем не составного типа. В этом случае в запросе sql  будет настроена связь левым соединением.

**3.** Условия в секции ГДЕ накладываются **до** секции с группировкой. Чтобы была возможность наложить отбор после того как как произошла группировка нужно использовать секцию **ИМЕЮЩЕ**. В секции где указываются поля с учетом приминения функции (сумма, количество и т.д.)



**4.** Если в запросе присутствуют итоги, то результатом запроса будет дерево значений.

**5. Виртуальная таблица** — это таблица которая содрежит в себе название регистра и плюс некоторый дополнительный идентификатор (срез последних, оброты и т. д.). Виртуальные таблицы не хранятся в базе данных, они получаются расчетным путем на основании физической таблицы.

**ВАЖНО:** Если нужно наложить условия для виртуальной таблицы, то нужно это делать в параметрах виртуальной таблицы, а не в секции **ГДЕ**.

**Разница между условиями в параметрах виртуальной таблицы и секцией ГДЕ:**

* Если условие накладывается в секции **ГДЕ**, то сначала будет получена таблица по всем абсолютно измерениям и только потом будет наложен фильтр.



* Если условие накладывается в виртуальной таблице, то при получении данных результат сразу будет отфильтрованный.



Другими словами, если используются параметры виртуальной табицы, то фильтр будет срабатывать максимально рано и за счет этого лишние данные прочитаны не будут.

**Виртуальные таблицы регистра сведений:**

* Если регистр сведений непериодический, то у такого регистра виртуальных таблиц нет.
* Если регистр периодический, то для него всегда доступны две виртуальные таблицы:

**СрезПервых() —** предназначена для получения значений ресурсов на будущую дату, ближайшую к выбранной дате среза.

**СрезПоследних()** — предназначена для получения значений ресурсов на прошлую дату, ближайшую к выбранной дате среза.

**Виртуальные таблицы регистра накопления:**

* Для регистра накопления виртуальные таблицы есть всегда, но количество этих таблицы зависит от того какой вид регистра выбран.
* Для регистра вида остаткиесть три виртуальные таблицы: **Обороты**, **Остатки** и **ОстаткиИОбороты**.
* Для регистра вида обороты есть только одна виртуальная таблица: Обороты.

**Виртуальные таблицы регистра накопления используются для:**

* **Остатки()** — используется для получения значений остатка ресурсов в разрезе измерений **на дату**.

Возможные параметры виртуальной таблицы типа остатки: **период**, **условие**.

* **Обороты()** — используется для получения значений оборота ресурсов в разрезе измерений **за период**.

Возможные параметры виртуальной таблицы типа обороты:

**НачалоПериода**

**КонецПериода**

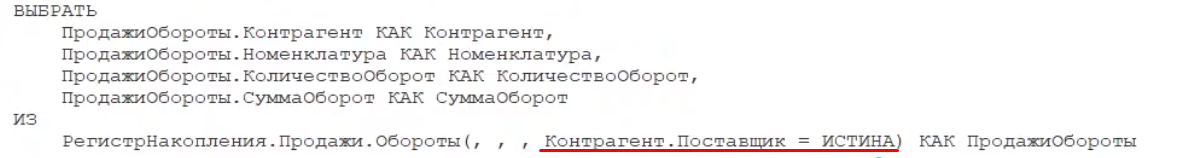
**Условие**

**Периодичность** — позволяет получить дополнительный разворот по периодам. Тоесть если стоит периодичность год, то обороты будут получатся в разрезе года.

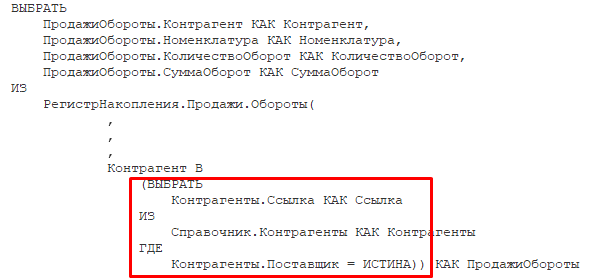
* **ОстаткиИОбороты()** — используется для того, чтобы получать информацию как по остаткам так и по оборотам. Используется при построении отчетов типа оборотно-сальдовая-ведомость, когда нужно получить за выбранный период начальный остаток, обороты за период(приход расход) и конечный остаток.

**6. Наложение фильтров по реквизитам измерения(разыменование) в виртуальнх таблицах делать нельзя !** Это может привести к долгому выполнению запроса. Вместо этого нужно использовать отбор с использованием вложенно го запроса.

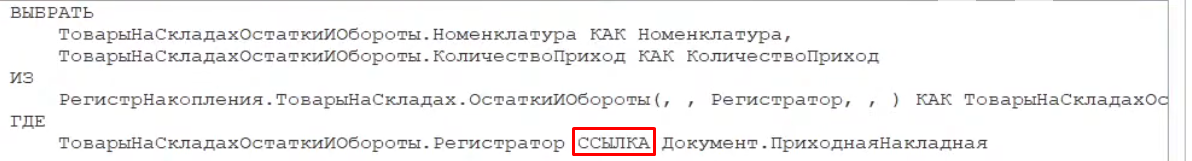
**Так делать нельзя:**



**Нужно делать так:**



**7.** Чтобы указать отбор по типу поля нужно использовать оператор **ССЫЛКА**.



**8. Вложенный запрос (подзапрос)** — предназначен для использования результата одного запроса внутри другого запроса.

Пример:

Получить остатки по номенклатуре конкретного документа (есть только ссылка на документ).

Внутри запроса, который получает остатки, нужно использовать данные запроса, который получает данные из табличной части документа.

**9. Соединения таблиц**

* **Левое соединение:**

Попадают все записи из левой таблицы, а из правой только те у которых есть сопоставление

В итоговой таблице будут только те записи которые были в левой.

* **Правое соединение:**

тоже самое что и левое, но наоборот.

* **Внутреннее соединение:**

Получаем только те записи для которых выполняется условие соединения.

Тоесть это левое соединение но без значений null.

* **Полное соединение:**

Получаем все записи из первой таблицы и все записи из второй таблицы.

Если между какими то записми не будет найдено соответствие, то там будут значения null.

* **Безусловное соединение (декартовой соединение):**

Получаем все возможные комбинации из левой и правой таблицы. Тоесть с первой строкой левой таблицы будут соотнесены все строки из правой таблицы, потом со второй строкой из левой таблицы будут соотнесены опять все строки из правой таблицы и т. д. Это соединение настравивается вообще без связей.

**10.** Проверять значение на **NULL** нужно с помощью **ЕСТЬ NULL**. Так как сравнивать значение null с null будет всегда ложь. Использовать сравнение через знак «=» не правильно.

**11.** Разница между **ОБЪЕДИНИТЬ** и **ОБЪЕДИНИТЬ ВСЕ**:

* **ОБЪЕДИНИТЬ ВСЕ** — объединить результаты запросов, в итоговой таблице дубли строк не удалять (не выполнять группировку).
* **ОБЪЕДИНИТЬ** — объединить результаты запросов и удалить дубли строк в итоговой таблице (выполнить группировку). Этот вариант более трудоемкий для СУБД так как здсь выполняется еще одно дейсвтие. На прктике первый вариант используется чаще.

**12. Временные таблицы.**

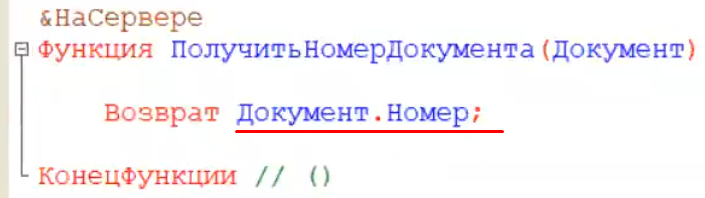
* Предназначение: временное хранение данных при выполнении запроса.
* Позволяют сделать текст сложных запросов более легким для восприятия.
* Создаются по умолчанию в оперативной памяти. СУБД может при необходимости помещать временную таблицу в служебную БД «TempDB».
* Чтобы уничтожить временную таблицу нужно создать еще один запрос и разместить его после той временной таблицы которую мы хотим уничтожить и на вкладке «Дополнительно» указать тип запроса «унчистожение...» и указать имя уничтожаемой таблицы. Уничтожение временной таблицы позволяет более грамотно использовать ресурсы компьютера. При уничтожении временной таблицы она удаляется из оперативной памяти.
* В идеале временные таблицы не должны содержать очень много строк. От этого напрямую зависит производительность запроса.

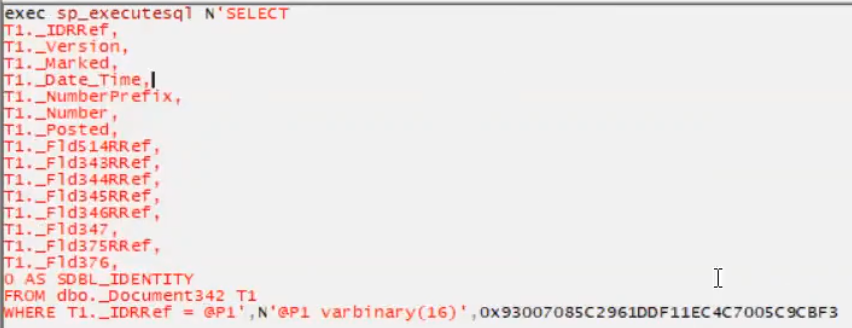
**ВАЖНО:** Если стоит задача получать данные из виртуальной таблицы и из какой то другой таблицы, то соединять их плохая идея ! Более оптимально прочитать данные из виртуальной таблицы с нужным отбором и поместить их во временную таблицу, а потом соединяться со временной таблицей. ТОЕСТЬ СОЕДИНЯТЬСЯ С ВИРТУАЛЬНЫМИ ТАБЛИЦАМИ ТАК И С ПОДЗАПРОСАМИ ЭТО ПЛОХО !

**13. Метод ВыполнитьПакетСПромежуточнымиДанными()** — если в запросе есть временные таблицы, то этот метод позвоялет получить их результат.

**14.** Когда мы обращаемся через точку к реквизиту, то на самом делое происходит **sql** запрос в котором выбираются все реквизиты. Поэтому если нам нужно значение реквизита лучше использовать запрос и явно указывать какое поле нужно получить.

Не правильно:





Правильно:

