SOFTPOINT

АУДИТ ПРОИЗВОДИТЕЛЬНОСТИ

СВОДНЫЙ ОТЧЁТ

Россия 107140, г. Москва, 3-й Красносельскийпер., д.21, стр.1, офис 305
Тел.: +7(495) 543-74-02· www.softpoint.ru ·Partner@softpoint.ru

Оглавление

1. Введение

Цель работ

Выявление узких мест в работе информационной системы компании Заказчика, развёрнутой на сервере 1С8-6, на предмет производительности, с точки зрения серверной и сетевой инфраструктуры, а также в разрезе используемого программного обеспечения информационной системы.

Состав работ

- Аудит аппаратных, программных ресурсов и архитектуры;
- Аудит программной части;
- Составление списка рекомендаций;
- Подготовка комплексного итогового отчёта.

Порядок проведения работ

- Установка программного комплекса PERFEXPERT для сбора статистики о производительности;
- Сбор и анализ статистики о производительности;
- Анализ серверной архитектуры в целом и анализ текущих настроек;
- Анализ узких мест с точки зрения программной части информационной системы
- Предоставление отчёта по аудиту информационной системы.

Основные выводы (кратко)

Главной проблемой текущей информационной системой является ожидание на избыточных блокировках. Частично в этом виноваты внутренние механизмы СУБД, частично — неоптимальные запросы, настройки конфигурации и код 1С.

Кроме того, во время работы пользователи часто сталкиваются с отменой транзакции записи документов и справочников. Такие ошибки негативно сказываются на общей производительности системы.

Также существуют неоптимальные настройки СУБД и неоптимальные запросы, негативно влияющие на производительность.

2. Основные выводы

2.1 Общий вывод

Наиболее критичной на данный момент видится проблема с блокировками. Большая частота обращений к таблицам регистрации изменений объектов вызывает эскалацию блокировок, что, в свою очередь, тормозит работу пользователей. К тому же, автоматический режим работы с блокировками, установленный в конфигурации 1С, вызывает избыточный уровень изоляции транзакций. Рекомендуется отключить эскалацию блокировок, а также перейти на управляемые или гибкие блокировки. Это позволит увеличить параллельность работы пользователей.

Кроме того, в СУБД включено параллельное выполнение частей запросов, что в итоге приводит только к увеличению времени выполнения запроса и излишней загрузке процессора. Рекомендуется отключить эту возможность или установить большой порог для использования параллелизма.

Во время работы с 1С пользователи сталкиваются с большим числом отмен транзакций. Такие операции негативно влияют на производительность. Рекомендуется пересмотреть логику работы пользователей с программой, чтобы уменьшить число отменённых транзакций.

Также стоит обратить внимание на следующие рекомендации:

- Указать минимальный и максимальный объём памяти, доступной MS SQL
- Провести оптимизацию кода обработки проведения наиболее используемых документов.

Кроме того, в системе существуют неоптимальные запросы, создающие паразитную нагрузку на аппаратные ресурсы. Оптимизация таких запросов позволит увеличить потенциал роста нагрузки на систему.

2.2 Заключение по программной части СУБД MS SQL Server

Важной проблемой текущей системы, влияющей на производительность, являются избыточные блокировки. Статистика по событиям блокировок за время наблюдения (С 4 по 11 ноября 2013г.) показывает, что пользователи столкнулись с 2 999 ожиданиями на блокировках. Общее время ожидания составило 3,6 часа.

Таблица 1. Статистика событий блокировок

№	Тип события	Кол. запросов	сумма длительность	среднее длительность	% доля длительность	макс. длительность
1	Lock:Acquired	2 856	2ч 58м	3,74с	79,76%	19,59с
2	Lock:Cancel	10	50,52с	5,05с	0,38%	13,64с
3	Lock:Timeout	133	44м 20с	20,00с	19,86%	20,02с
	Сумма	2 999	3ч 43м		100,00%	

В поисках причин избыточных блокировок можно было бы предположить, что на количество блокировок влияет состояние индексов баз данных. Но на самом деле это не так: ежедневное перестроение индексов позволяет поддерживать показатели качества индексов на высоком уровне.

Если проанализировать статистику по объектам, на которых возникают блокировки, становится заметно, что около половины всех блокировок — табличные. Таблица полностью блокируется одной транзакцией и до конца её (транзакции) выполнения, никто

больше не может получить доступ к этой таблице. Естественно, такое поведение СУБД резко снижает параллельность работы пользователей — получается, что в определённые моменты работа с объектами базы данных становится «однопользовательской».

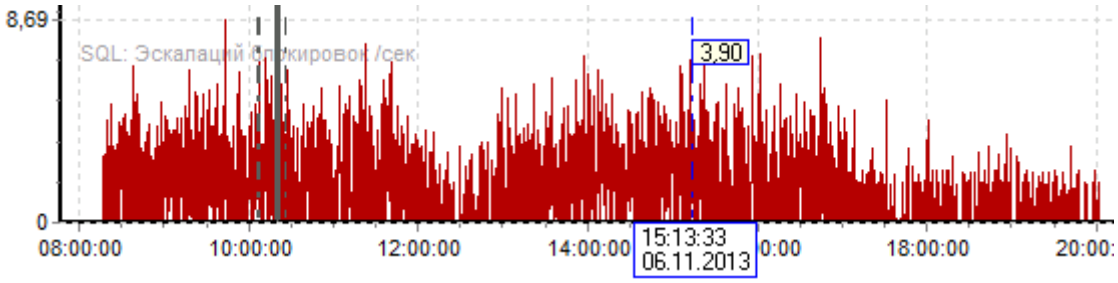


Рисунок 1. График количества эскалаций блокировок в течение рабочего дня

График возникновения эскалаций блокировок подтверждает наблюдение: в течение всего рабочего дня наблюдаются случаи повышения уровня блокировок. Надо заметить, что эскалация блокировок возникает в случае, когда движок базы данных видит большое количество обращений к одной таблице. СУБД решает заблокировать её всю вместо того, чтобы управлять сотнями мелких блокировок на смежные страницы данных. При запросе большого числа строк такой подход позволяет экономить ресурсы системы (каким бы мелким объектом ни была блокировка, на её обслуживание все равно тратится часть процессорной мощности и оперативной памяти).

В ситуации, когда в системе работает большое количество пользователей, и нет дефицита аппаратных ресурсов, можно произвести размен ресурсов на параллельность работы — отключить эскалацию блокировок. При этом увеличится количество мелких блокировок, потребуется дополнительная оперативная память для их обслуживания, но, в то же время, пользователи перестанут сталкиваться с табличными блокировками — а значит, больше операций может быть проведено параллельно. **Отключить эскалацию блокировок можно, включив флаг трассировки 1211.**

Ещё более очевидным ущерб от эскалации блокировок становится, если обратить внимание на то, какие таблицы чаще всего оказываются целью блокировки. При просмотре 10 таблиц, на которых происходит наибольшее количество блокировок, становится заметно, что сильно нагружены таблицы регистрации изменений объектов (таблицы с суффиксом ChangeRec). В эти таблицы записываются ссылки на объекты, которые участвуют в обмене данными и чьи изменения необходимо передать в другие узлы. В обычных условиях такие таблицы не создают проблем с производительностью — записываемые данные (по сути — ссылки на разные объекты информационной базы) не пересекаются и у пользователей нет возможности заблокировать друг друга. Но вспомним, что говорилось ранее про эскалации блокировок — в данном случае СУБД видит большое количество запросов на изменение таблицы регистрации, не смотря на то, что эти запросы не пересекаются, поднимает уровень блокировки одной транзакции до табличного. В результате один пользователь блокирует всю таблицу и все остальные транзакции вынуждены ждать ее освобождения.

Таблица 2. Объекты, на которых происходит наибольшее количество блокировок

№	Объект	Кол. запросов	сумма длительность	среднее длительность	% доля длительность	макс. длительность
1	0	1 297	1ч 18м	3,61с	34,97%	20,02с
2	_BusinessProcess8775	783	1ч 13м	5,63с	32,92%	20,02с
3	_Task4401	454	41м 26с	5,47с	18,56%	20,02с

4	_DocumentChangeRec2178	232	14м 35с	3,77с	6,54%	20,02с
5	_ReferenceChangeRec3028	126	7м 34с	3,60с	3,39%	20,02с
6	_BusinessProcess8086	31	2м 53с	5,59с	1,29%	20,02с
7	_InfoRegChangeRec3065	19	0,53с	0,03с	0,00%	0,13с
8	_DocumentChangeRec4540	17	55,81с	3,28с	0,42%	11,89с
9	_ReferenceChangeRec2177	16	1м 14с	4,68с	0,56%	17,75с
10	_Document54	6	1м 16с	12,82с	0,57%	20,00с

Свой вклад в увеличение длительности блокировок вносит автоматический режим блокировок, включённый в конфигурации. При этом на уровне базы данных используются уровни изоляции транзакций «REAPEATABLE READ» и «SERIALIZABLE», что негативно сказывается на параллельности работы пользователей. Рассмотрим один из запросов (рис. 2): видно, что таблицы выбираются с высоким уровнем изоляции (отмечено красным). В результате, СУБД вынуждена простаивать в ожидании ресурсов: процессорное время (читай: время выполнения операций с данными) в разы меньше, чем общее время выполнения запроса. **Рекомендуется перейти на управляемые блокировки или вообще внедрить технологию гибких блокировок.** Это позволит увеличить параллельность работы пользователей и избежать избыточных блокировок.

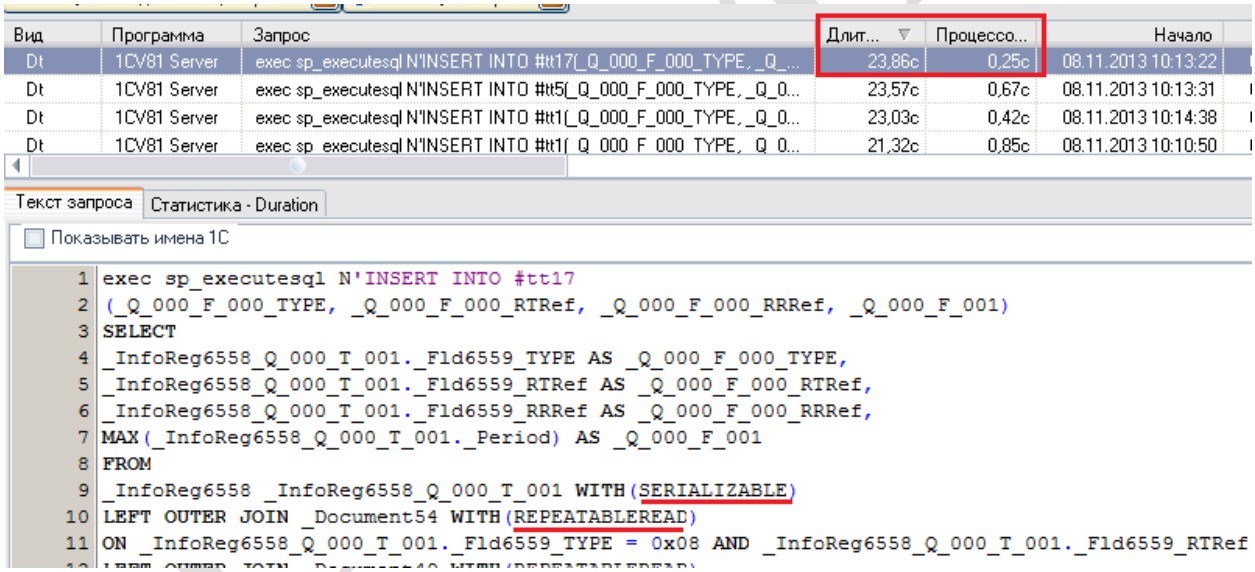


Рисунок 2. Высокий уровень изоляции транзакций при выполнении запросов

При просмотре деревьев блокировок обращают на себя внимание записи с типом ожидания «CXPACKET». Такие ожидания означают, что выполнение запроса было разбито на несколько частей и в данный момент одна из частей выполнилась, но не может отпустить ресурсы процессора и оперативной памяти, пока не дождётся окончания выполнения остальных частей.

Для OLTP-систем, к которым относят и 1С, не рекомендуется включать параллельное выполнение запросов: характер выполняющихся запросов и профиль нагрузки таких систем не позволяет добиться большого выигрыша от параллелизма. В то же время, накладные расходы на распараллеливание, а затем сборку запроса в одно целое, а также ожидание «опаздывающих» потоков — заметно сказываются на производительности системы. С другой стороны, формирование тяжёлых «аналитических» отчётов несомненно выигрывает от включённого параллелизма: время выполнения сложных запросов может значительно уменьшиться.

Заметно, что в системе уже были попытки оптимизировать параллельное выполнение запросов — значение параметра «max degree of parallelism» отличается от значения по умолчанию. Видимо, такое значение продиктовано необходимостью периодически формировать тяжёлые отчёты. С другой стороны, по данным мониторинга видно, что из-за такой настройки тратятся избыточные ресурсы при проведении документов. Рекомендуется уменьшить значение параметра «max degree of parallelism» до 1 или 2. Если же после этого произойдёт сильное увеличение времени формирования тяжёлых запросов, можно вернуться к 6 потокам выполнения запросов и одновременно повысить порог включения параллелизма («Cost threshold for parallelism»). Конкретное значение порога должно подбираться эмпирически — можно начать с увеличения в 5-10 раз и, сверяясь со статистикой выполнения запросов, добиться такого значения, при котором параллелизм будет включаться только во время выполнения тяжёлых аналитических запросов.

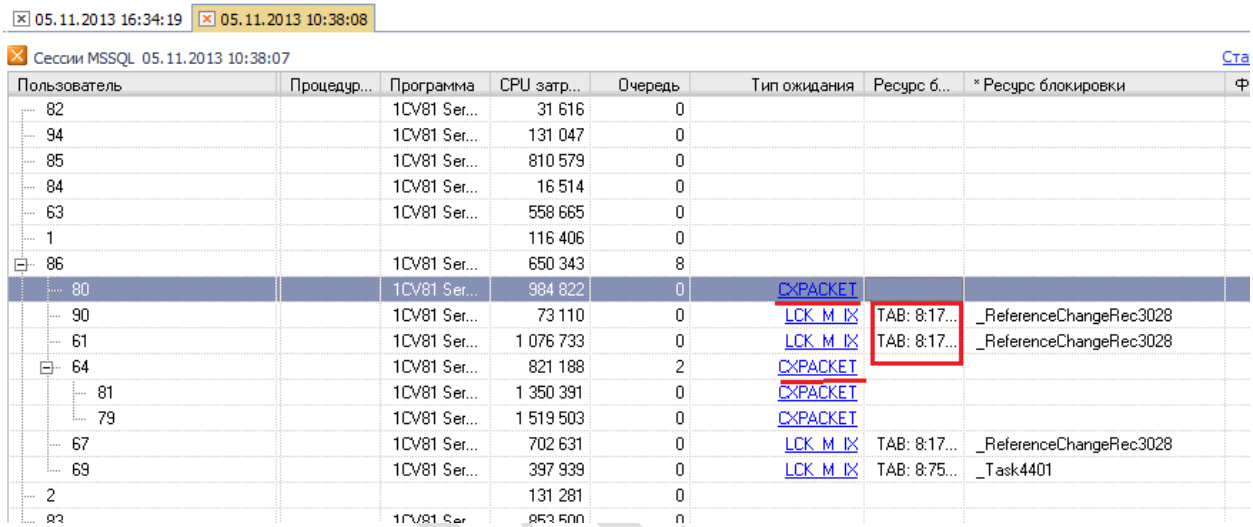


Рисунок 3. Типичное дерево блокировок: пользователь с подключением номер 86 заблокировал 8 других пользователей

График ожидаемого срока жизни страниц в памяти непрерывно растёт в течение всего дня, обрываясь только ночью, в момент выполнения регламентных операций по обслуживанию базы данных. Это говорит о том, что во время работы пользователей большая часть данных находится в быстром кэше в оперативной памяти, что позволяет СУБД быстро выполнять выборки данных.

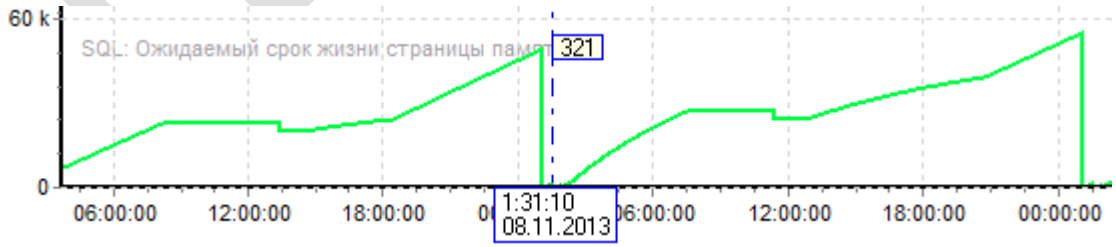


Рисунок 4. Изменение ожидаемого срока жизни страниц в памяти в течение рабочего дня

Рекомендуется задать минимальный объём памяти, используемый СУБД (параметр min server memory). **Microsoft рекомендует оставлять небольшую вилку между значениями максимального и минимального объёма доступной памяти.** При этом сервер работает в режиме динамически выделяемой памяти, что является предпочтительным с точки зрения его внутренних механизмов, но в то же время не может столкнуться с нехваткой оперативной памяти, т.к. просто не освобождает память ниже установленного предела.

Как и в случае с сервером SUBD01, необходимость такой настройки легко иллюстрируется сравнением графиков свободной RAM и памяти, потребляемой СУБД. Видно, что они практически являются зеркальным отражением друг друга, т.е. на сервере нет других задач, которые могут потреблять свободную оперативную память. А если память и так достанется SQL Server, лучше заранее позволить ему занять определённый объем памяти и не тратить потом ресурсы системы на её (памяти) перераспределение.

Также рекомендуется ограничить максимальный размер памяти, потребляемой SQL Server. Не смотря на то, что за время наблюдения СУБД ни разу не смогла занять столько памяти, чтобы создать проблемы операционной системе, всё равно стоит учитывать такую возможность и задать ограничение таки образом, чтобы оставить ОС как минимум 4Гб оперативной памяти.

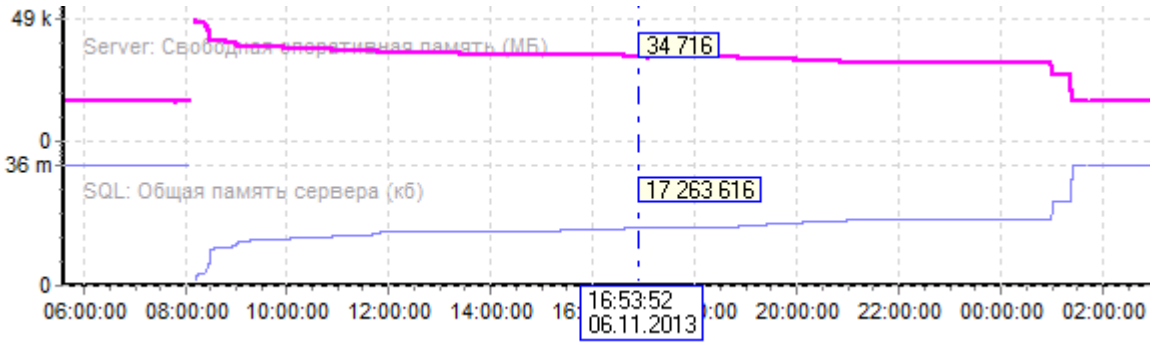


Рисунок 5. Вверху — свободная оперативная память; внизу — память, используемая MS SQL Server

Кроме того, в системе присутствуют неоптимальные запросы, генерирующие паразитную нагрузку на процессор и подсистему ввода-вывода. Подробнее примеры таких запросов будут рассмотрены в разделе 3, сейчас же просто отметим, что оптимизация таких запросов позволит увеличить потенциал роста нагрузки на текущую систему.

2.3 Заключение по программной части 1С

Статистика по наиболее долгим операциям 1С за период с 4 по 11 ноября показывает, что обработка проведения большинства документов укладывается в промежуток 5 – 15 секунд. С другой стороны, большое количество вводимых документов приводит к тому, что ускорение обработки проведения даже на одну секунду выливается в ощутимую экономию времени для операторов. Рекомендуется в первую очередь обратить внимание на документы «Карточка договора» и «Технические условия ТП».

Таблица 3. Наиболее тяжёлые операции 1С

№	Модуль 1С	Кол-во	Сумма длит.	Сред. длит.	% длит.	Макс. длит.
1	Документ.КарточкаДоговора	827	1ч 2м	4,52с	58,84%	31,75с
2	Документ.ТехническиеУсловияТП	277	10м 43с	2,32с	10,12%	31,23с
3	Документ.КарточкаДоговора.Форма.ФормаДокумента	52	9м 14с	10,66с	8,72%	38,03с
4	Документ.РегистрацияЗаявкиНаРасходованиеДенежныхСредств	134	6м 9с	2,76с	5,81%	22,17с
5	Документ.Спецификация	255	4м 24с	1,04с	4,16%	8,98с
6	Документ.ТехническиеУсловияТП.Форма.ФормаДокумента	78	3м 58с	3,06с	3,76%	22,97с
7	Документ.ЗаявкаНаОплату	170	3м 43с	1,31с	3,51%	13,35с
8	Документ.ОбращениеПотребителяОвыполненииТУ	134	2м 32с	1,14с	2,41%	19,56с

9	Документ.РасчетСтоимостиТехнологическогоПрисоединения	149	38,75с	0,26с	0,61%	6,55с
10	Документ.АктТехнологическогоПрисоединения	48	36,29с	0,76с	0,57%	2,34с
11	Документ.ЗаявкаНаОплату.Форма.ФормаДокумента	10	30,52с	3,05с	0,48%	20,83с
12	Отчет.УниверсальныйРеестрЗаявокТП	16	22,59с	1,41с	0,36%	4,01с
13	Документ.ОбращениеПотребителяОвыполненииТУ.Форма.ФормаДокумента	1	20,48с	20,48с	0,32%	20,48с
14	Документ.РасчетПредварительнойСтоимостиДоговораТП	75	20,09с	0,27с	0,32%	0,70с
15	Документ.Спецификация.Форма.ФормаСпецификации	2	0,45с	0,23с	0,01%	0,30с
16	МодульПриложения	3	0,39с	0,13с	0,01%	0,19с
	Сумма	2 231	1ч 45м		100,00%	

При этом детальный анализ запросов, выполняемых внутри транзакций проведения документов, показывает, что непосредственно обращения к СУБД составляют малую часть от общего времени проведения/формирования отчётов. С другой стороны, заметно, что на проведение документов влияют ожидания на блокировках — если уменьшить их количество, скорость проведения документов увеличится.

В приложении 2 приведена статистика по ошибкам, с которыми чаще всего сталкиваются пользователи информационной системы. Заметно большое количество ошибок из-за отмены транзакций (первые 2 строки статистики, которые в сумме дают около 40% общего количества ошибок). Эти ошибки означают, что транзакция была прервана и произошла отмена сделанных изменений (rollback). Отмена изменений — одна из самых тяжёлых для СУБД операций в плане производительности. MS SQL оптимизирован из расчёта, что большинство транзакций выполняется успешно, поэтому каждая отмена транзакции выливается в сложную последовательность изменений в файлах данных. **Рекомендуется пересмотреть логику работы пользователей и минимизировать количество отмен транзакций.**

2.4. Заключение по аппаратным ресурсам

За время наблюдения с 4 по 11 ноября были получены следующие показатели :

- Средняя нагрузка CPU: 25%, максимальная: 75%
- Средняя длина очереди процессора: 0,1; максимальная: 6
- Средний объем свободной оперативной памяти: 24 512 Мб
- Средняя длина очереди к диску: 0,26; максимум: 1722

Подробные графики наиболее важных показателей приведены в приложении 1.

Видно, что текущие аппаратные средства справляются с обслуживанием информационной системы. Наиболее критичным ресурсом видится процессор, но даже у него есть достаточный запас мощности, чтобы выдержать рост нагрузки в 10-25%.

2.5 Регламентные операции

Для поддержания должного уровня производительности SQL сервера рекомендуется выполнять регламентные работы со следующей периодичностью:

- Перестроение индексов базы данных. Перестроение индексов имеет очень большое влияние на общую производительность системы. Рекомендуется выполнять перестроение индексов раз в неделю, при необходимости — более часто, до 1 раза в сутки. Скрипт для MS SQL приведён в приложении 3.

- Автоматическое обновление статистики. При работе с базой данных с включённой опцией автоматического обновления статистики при каждом добавлении изменении или удалении данных происходит соответствующая перестройка статистики. Тем не менее, иногда статистика, особенно в случае массированных действий, не соответствует действительной. Рекомендуется работы по пересчёту статистики настроить на проведение в автоматическом режиме в нерабочее время. Рекомендуется обновлять статистику раз в день.
  UPDATE STATISTICS table
  Выполняется в нерабочее время один раз в день в контексте БД 1С.

- Очистка кэша выполнения запросов - DBCC FREEPROCCACHE (каждый день в 6:00)
  Выполняется в нерабочее время один раз в день в контексте БД master.

Выполнение данных рекомендаций позволит сохранить показатели качества работы оптимизированной системы, несмотря на увеличение количества обрабатываемых данных.

3. Причинно-следственные связи

3.1. Неоптимальные запросы

Как было сказано ранее, в системе присутствуют неоптимальные запросы, которые не позволяют движку СУБД выбирать данные с максимальной эффективностью. Рассмотрим подробнее основные ошибки в этих запросах.

Обратим внимание на запрос, наиболее нагружающий процессор (статистика по наиболее тяжёлым запросам приведена в приложении 4; полный текст разбираемого запроса — в приложении 5). Видно, что задача запроса — отфильтровать содержимое таблицы _Task4401 (таблица задач пользователя), причём значение, по которому происходит отбор, может храниться в одной из шести других таблиц. Собственно, основная сложность данного запроса состоит в том, что для выполнения отбора необходимо произвести соединение с шестью разными таблицами, в результате объём данных для обработки катастрофически возрастает. Можно, конечно, рассчитывать на то, что оптимизатор запросов SQL Server разберётся в ситуации и сначала отфильтрует те «дополнительные» таблицы, а потом уже будет выполнять соединение с основной таблицей задач. В идеальных условиях так и произойдёт. Но уповать на алгоритмы оптимизатора — плохое архитектурное решение, слишком много факторов влияет на то, какой способ выполнения запроса может быть выбран. Устаревшая статистика, сильная фрагментация индексов, недостаток оперативной памяти — и запрос уже выполняется неоптимальным способом, затягивая время ожидания пользователя и генерируя дополнительную нагрузку на ресурсы сервера. Поэтому лучше изначально строить конструкцию запроса так, чтобы избежать неоптимального выполнения.

```
WHERE CASE
        WHEN _Task4401_Q_000_T_001._BusinessProcess_TYPE = 0x08
            AND _Task4401_Q_000_T_001._BusinessProcess_RTRef = 0x0000135C
            THEN _BusinessProcess4956._Fld4962RRef
        WHEN _Task4401_Q_000_T_001._BusinessProcess_TYPE = 0x08
            AND _Task4401_Q_000_T_001._BusinessProcess_RTRef = 0x00001F96
            THEN _BusinessProcess8086._Fld8099RRef
        WHEN _Task4401_Q_000_T_001._BusinessProcess_TYPE = 0x08
            AND _Task4401_Q_000_T_001._BusinessProcess_RTRef = 0x0000182F
            THEN _BusinessProcess6191._Fld6220RRef
        WHEN _Task4401_Q_000_T_001._BusinessProcess_TYPE = 0x08
            AND _Task4401_Q_000_T_001._BusinessProcess_RTRef = 0x0000112F
            THEN _BusinessProcess4399._Fld4465RRef
        WHEN _Task4401_Q_000_T_001._BusinessProcess_TYPE = 0x08
            AND _Task4401_Q_000_T_001._BusinessProcess_RTRef = 0x000023ED
            THEN _BusinessProcess9197._Fld9216RRef
        WHEN _Task4401_Q_000_T_001._BusinessProcess_TYPE = 0x08
            AND _Task4401_Q_000_T_001._BusinessProcess_RTRef = 0x00002247
            THEN _BusinessProcess8775._Fld8778RRef
```

Рисунок 6. Условие отбора задач пользователя. Видно, что в условии выбираются реквизиты из "правых" таблиц, которые, по сути, имеют один и тот же прикладной смысл

В данном случае, чтобы избежать большого количества соединений с вспомогательными таблицами, можно подумать о переносе информации, общей для этих таблиц, в таблицу задачи. При этом, возможно, произойдёт дублирование хранящейся информации, но, в то же время, процедура выборки данных значительно упростится.

Если обратить внимание на запросы, генерирующие наибольшее количество логических чтений (приложение 6), тоже становятся заметны неоптимальные моменты.

В приложении 7 приведён пример запроса из первой строчки списка наиболее тяжёлых запросов. Видно, что в условии отбора используется «тяжёлый» оператор LIKE, при том, что всё равно строка ищется по полному совпадению. Вообще поиск документа по номеру и дате, которые находятся внутри строки-описания, видится не самым эффективным решением. Можно задуматься о введении дополнительных полей справочника, которые помогут упорядочить информацию и оптимизировать поиск документов.

```
WHERE
_Reference2_Q_000_T_001._Fld116RRef = @P1
    AND _Reference2_Q_000_T_001._Fld115_TYPE = @P2
    AND _Reference2_Q_000_T_001._Fld115_S = @P3
    AND _Reference2_Q_000_T_001._Fld117
        LIKE N'' У вас на согласовании находится документ.
(указан в поле "Сопроводительный документ")

Окончание срока согласования: 19.02.2010 0:00:00
Дополнительное соглашение 082-000358 от 25.03.2008 11:54:05''',N'@P
истема согласования документов'
```

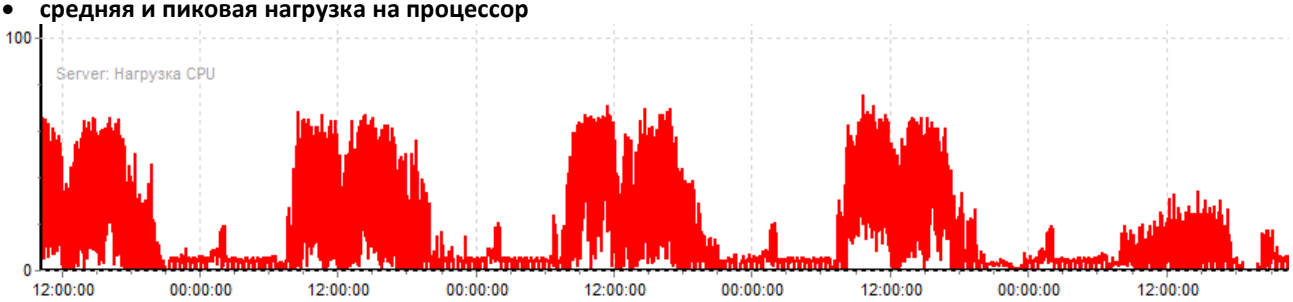
Рисунок 7. В условии поиска используется "тяжёлый" оператор LIKE, к тому же в строке поиска содержатся данные, которые целесообразно хранить в отдельных реквизитах

Приложение 1. Сводные данные о загрузке аппаратных ресурсов

Статистика работы сервера

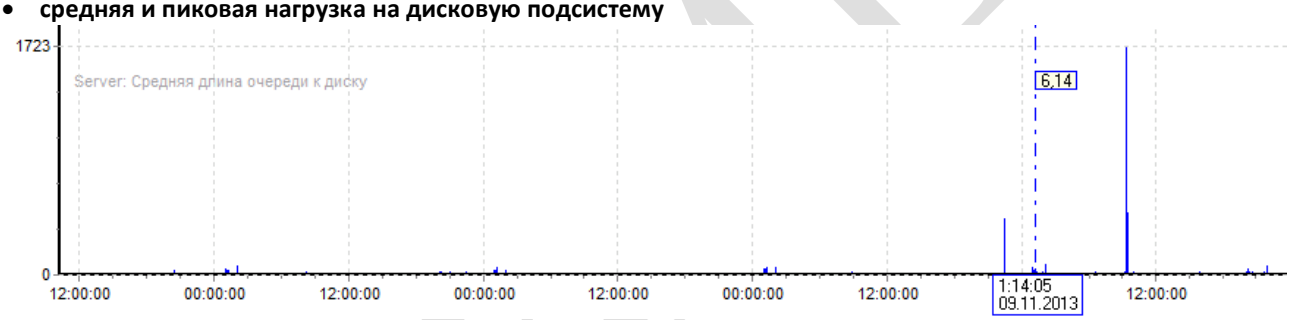
Период: с 4 по 11 ноября 2013 года.

- **средняя и пиковая нагрузка на процессор**



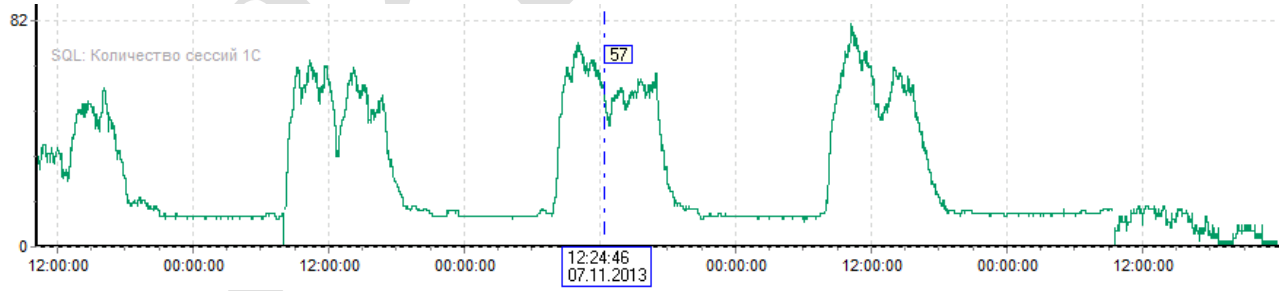
Средняя нагрузка на процессор: 25.054%
Пиковая нагрузка на процессор: 75.68 %

- **средняя и пиковая нагрузка на дисковую подсистему**



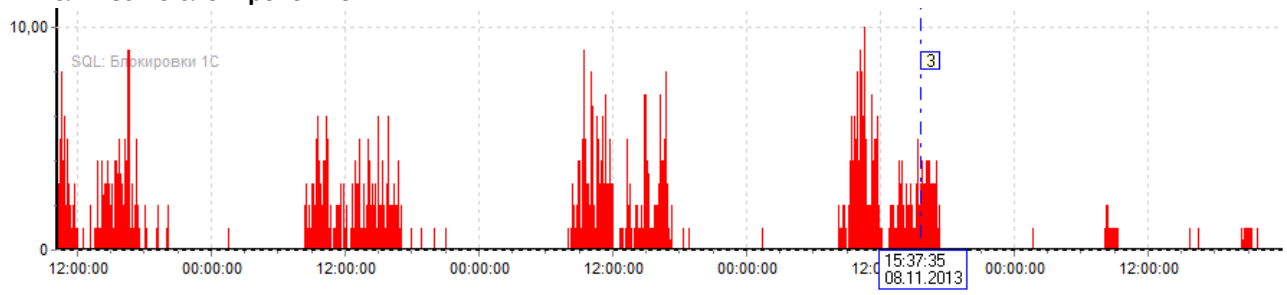
Средняя очередь к диску: 0.26.
Пиковая очередь к диску: 1722,20.

- **количество пользователей 1С.**



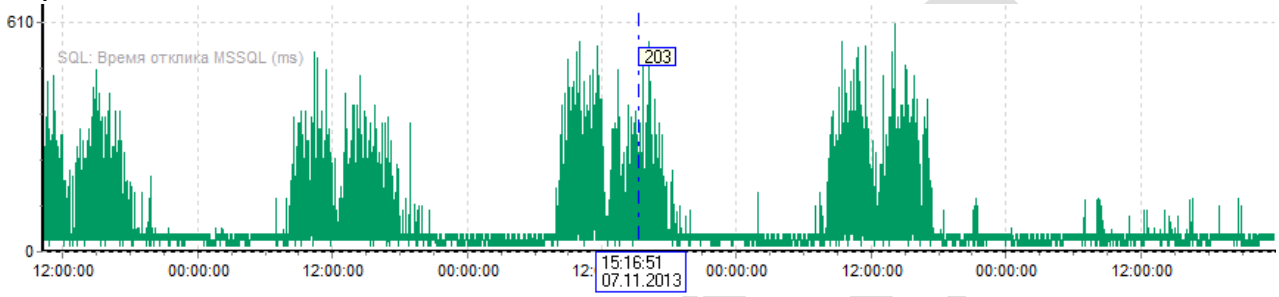
Среднее количество пользователей 1С: 29
Максимальное количество: 81

- **количество блокировок 1С**



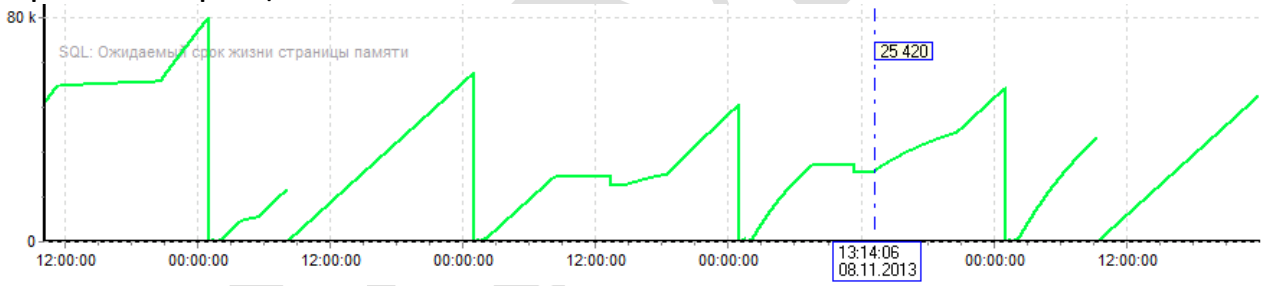
Среднее количество блокировок 1С: 0,24 (в секунду)
Максимальное количество блокировок 1С: 10

- **время отклика MS SQL**

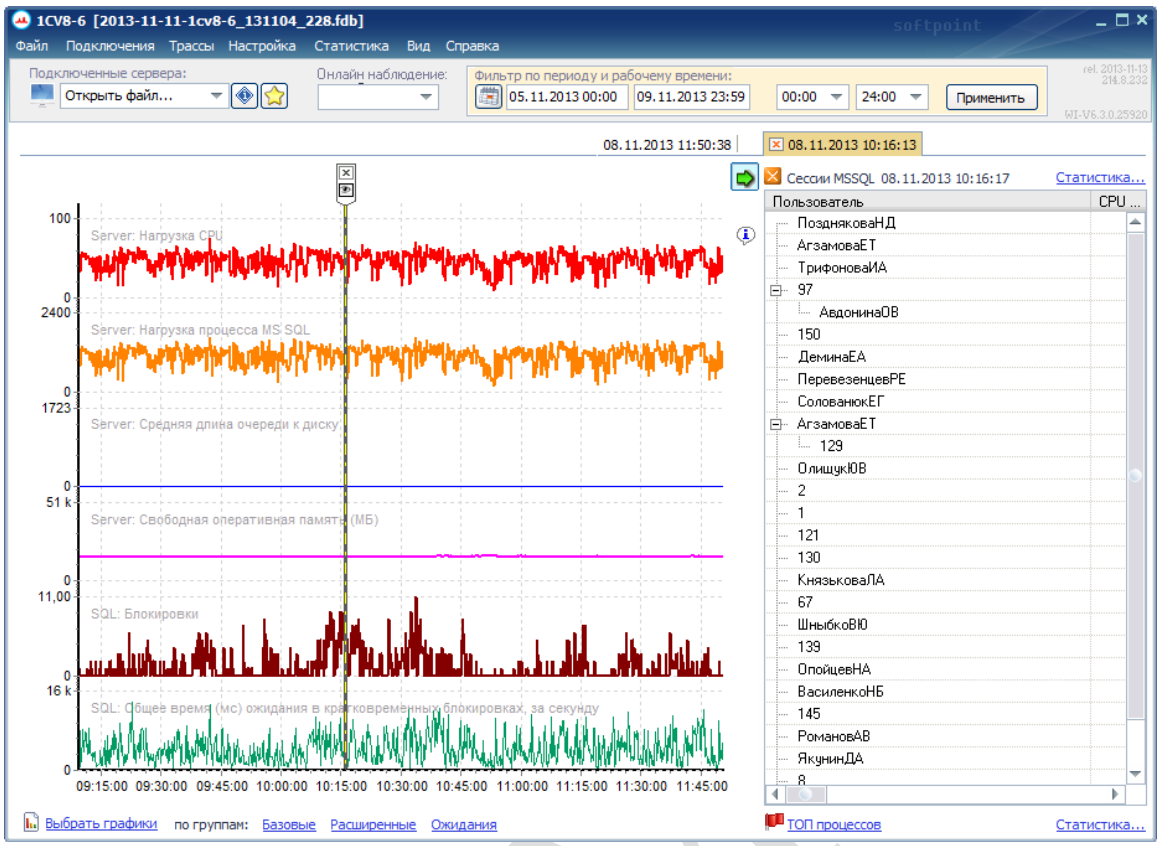


Среднее время отклика MS SQL: 81.73 мс
Максимальное время отклика: 609 мс

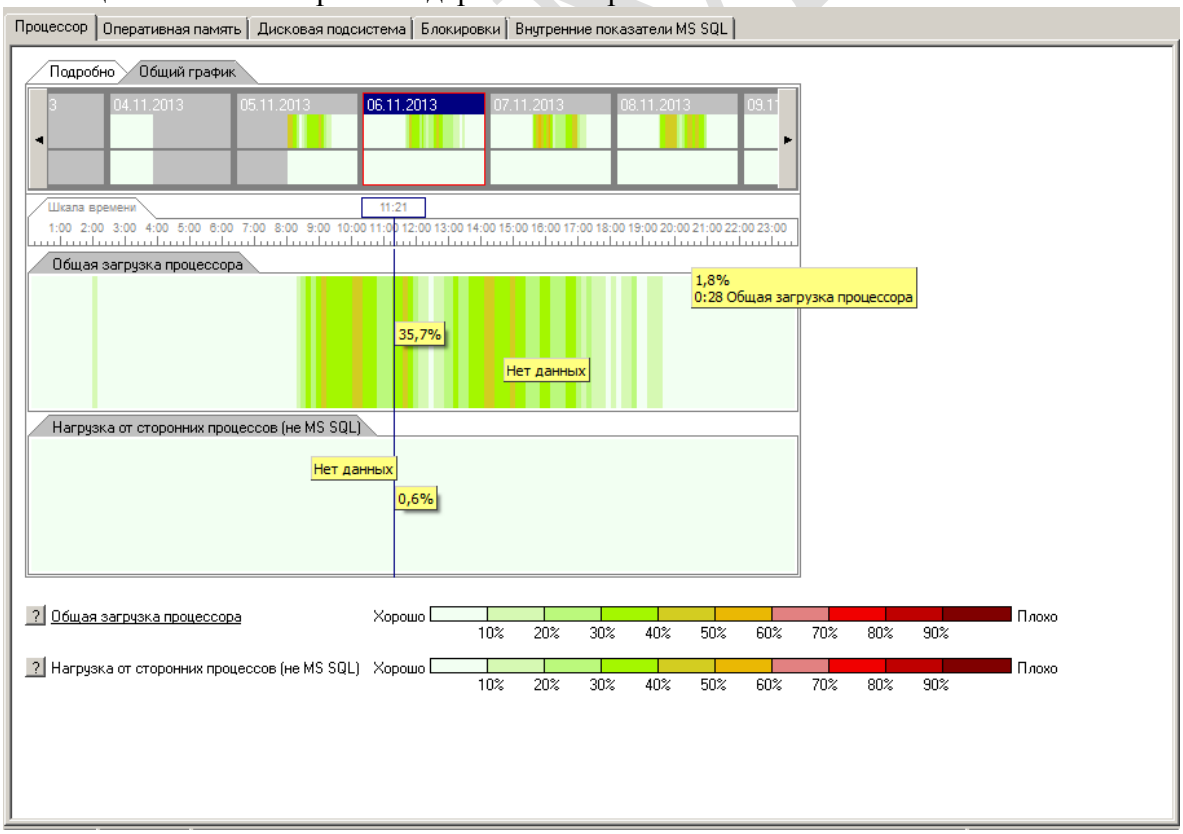
- **Время жизни страницы памяти**



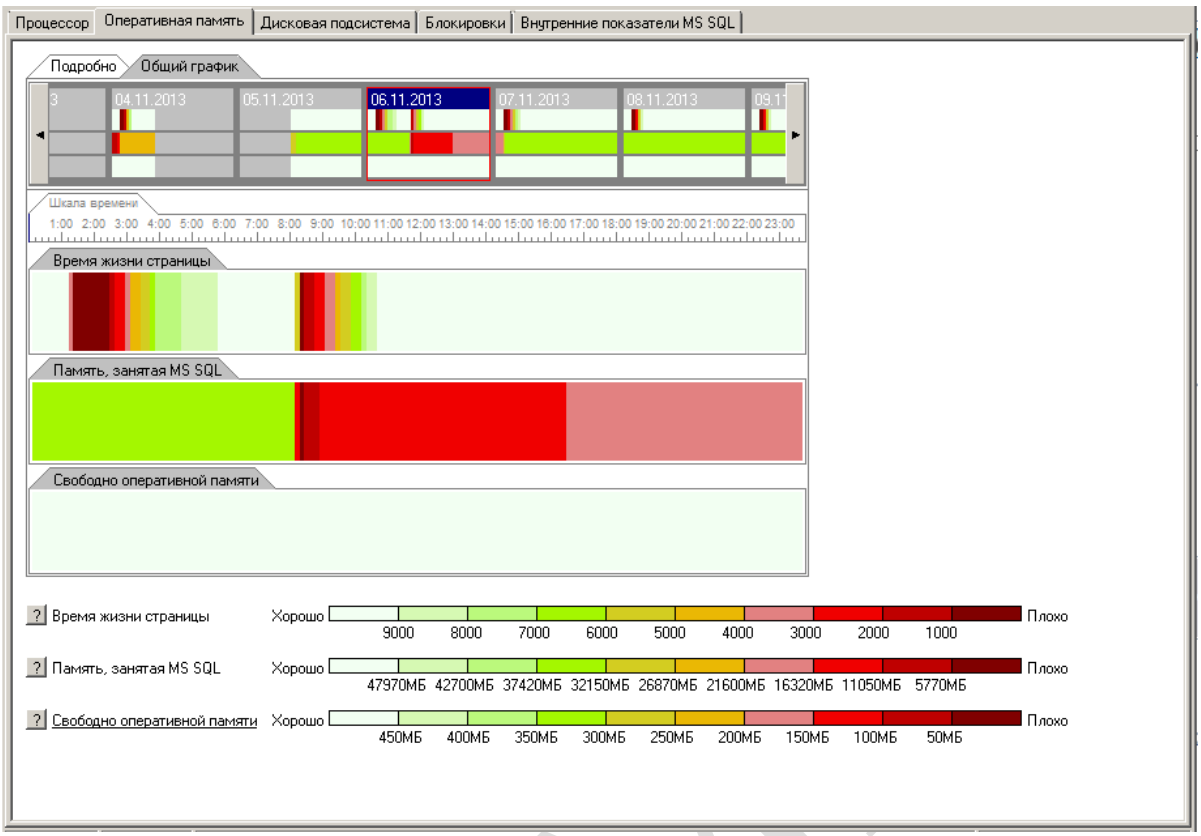
Среднее время жизни страницы: 29 462,05с
Максимальное время жизни страницы: 57 314с



Общее окно мониторинга и дерево блокировок.



По экспертной оценке аппаратных ресурсов хорошо видно, что CPU полностью не загружен, в течение всего рабочего дня.



По экспертной оценке памяти хорошо видно, что ситуация со временем жизни страницы в кеше неудовлетворительная.

Отчет по блокировкам

Сводная таблица по дням

Наименование	05.11.2013	06.11.2013	07.11.2013	08.11.2013	10.11.2013
Максимальное время ожидания блокировки, мс	20017	20000	686	20017	20000
Среднее время ожидания блокировки, мс	5005	4047	208	3637	20000
Средняя частота ожидания блокировок в час	142,00	83,80	7,00	495,20	2,00
Общее время ожидания блокировки за время измерения	1ч 58м 28сек	1ч 50м 18сек	2сек	4ч 33м 15сек	40сек
Количество ожиданий блокировки всего за время измерения	1420	1635	7	4507	2
Общее время неуспешных ожиданий	35м 02сек	14м 30сек		29м 02сек	40сек
Количество отмен ожидания блокировки всего за время измерения	111	45	0	92	2

Сводная таблица за все время

Наименование	За все время
Общее время ожидания блокировки за время измерения	8ч 22м 50сек
Общее время неуспешных ожиданий	1ч 19м 13сек

Наибольшее количество блокировок произошло 08.11.2013, в этот день возникло 4507 ожиданий блокировок, общее время ожидания пользователей составило 4ч 33м. При этом 10% (29м 01сек) времени ожидания пользователей закончилось неуспешно, т.е. пользовательская операция не была завершена. Необходимо отметить, что, как правило, после неуспешной блокировки пользователь не сразу повторяет операцию, т.е. время ожидания фактически выше.

1. Данные за 05.11.2013 (вторник).

Таблица 1.1. Сводная таблица.

Максимальное время ожидания блокировки, мс	20017
Среднее время ожидания блокировки, мс	5005
Средняя частота ожидания блокировок в час	142,00
Общее время ожидания блокировки за время измерения	1ч 58м 28сек
Количество ожиданий блокировки всего за время измерения	1420
Общее время неуспешных ожиданий	35м 02сек
Количество отмен ожидания блокировки всего за время измерения	111

Таблица 1.2. Ожидание блокировки в разрезе объектов баз данных.

	Объект БД	Длительность	Количество
	8.0	44м 55сек	619
	8._BusinessProcess8775	32м 31сек	325
	8._Task4401	17м 26сек	185
	8._DocumentChangeRec2178	10м 19сек	109
	8._ReferenceChangeRec3028	7м 35сек	126
	8._BusinessProcess8086	1м 47сек	15
	8._Document54	1м 17сек	5
	8._ReferenceChangeRec2177	1м 15сек	16
	8._DocumentChangeRec4540	41сек	9
0	8._AccumRegTurnovers4840	24сек	2
1	8._AccumRegChangeRec4560	20сек	1

Таблица 1.3. Отмена блокировки после ожидания в разрезе объектов баз данных.

	Объект БД	Длительность	Количество
	8.0	11м 02сек	39
	8._BusinessProcess8775	8м 41сек	26
	8._DocumentChangeRec2178	7м 21сек	22
	8._Task4401	4м 01сек	12
	8._BusinessProcess8086	1м 21сек	4
	8._ReferenceChangeRec3028	1м 01сек	3
	8._Document54	1м 00сек	3
	8._AccumRegChangeRec4560	20сек	1
	8._AccumRegTurnovers4840	20сек	1

Таблица 1.4. Ожидание блокировки в разрезе компьютеров.

	Компьютер	Длительность	Количество
	1CV85	1ч 58м 28сек	1420

Таблица 1.5. Ожидание блокировки в разрезе пользователей.

Пользователь	Длительность	Количество
1	1ч 43м 27сек	1247
2	3м 32сек	18
3	3м 28сек	40
4	1м 37сек	26
5	1м 29сек	9
6	1м 15сек	17
7	52сек	17
8	36сек	7
9	26сек	4
10	18сек	4
11	16сек	4
12	15сек	5
13	13сек	6
14	12сек	2
15	10сек	3
16	9сек	1
17	6сек	1
18	5сек	2
19	4сек	1
20	4сек	2
21	2сек	1
22	2сек	1

Таблица 1.6. Неуспешное ожидание блокировки в разрезе пользователей.

Пользователь	Длительность	Количество
1	29м 42сек	95
2	3м 01сек	9
3	1м 01сек	3
4	1м 00сек	3
5	21сек	1

Таблица 1.7. Отмена блокировки после ожидания в разрезе компьютеров.

	Компьютер	Длительность	Количество
	1CV85	35м 02сек	111

Таблица 1.8. Ожидание блокировки в разрезе видов блокировки.

	Вид блокировки	Длительность	Количество
	Прочие	1ч 58м 28сек	1420

Таблица 1.9. Отмена блокировки после ожидания в разрезе видов блокировки.

	Вид блокировки	Длительность	Количество
	Прочие	35м 02сек	111

2. Данные за 06.11.2013 (среда).

Таблица 2.1. Сводная таблица.

Максимальное время ожидания блокировки, мс	20000
Среднее время ожидания блокировки, мс	4047
Средняя частота ожидания блокировок в час	83,80
Общее время ожидания блокировки за время измерения	1ч 50м 18сек
Количество ожиданий блокировки всего за время измерения	1635
Общее время неуспешных ожиданий	14м 30сек
Количество отмен ожидания блокировки всего за время измерения	45

Таблица 2.2. Ожидание блокировки в разрезе объектов баз данных.

	Объект БД	Длительность	Количество
	8._BusinessProcess8775	41м 12сек	462
	8.0	37м 33сек	695
	8._Task4401	24м 04сек	271
	8._DocumentChangeRec2178	4м 26сек	126
	8._BusinessProcess8086	1м 07сек	16
	8._InfoRegChangeRec2970	24сек	2
	8._Document54	22сек	18
	8._ReferenceChangeRec326	20сек	1
	8._DocumentChangeRec4540	16сек	8
0	8._Document49	14сек	16
1	8._AccumRegTurnovers4840	12сек	1
2	8._InfoReg1536	6сек	1
3	8._Document2042	4сек	1
4	8._Document4353	2сек	1
5	8._AccumRegChangeRec4560	2сек	1

Таблица 2.3. Отмена блокировки после ожидания в разрезе объектов баз данных.

	Объект БД	Длительность	Количество
	8._BusinessProcess8775	8м 00сек	24
	8.0	2м 50сек	10
	8._Task4401	2м 20сек	7
	8._DocumentChangeRec2178	40сек	2
	8._InfoRegChangeRec2970	20сек	1
	8._ReferenceChangeRec326	20сек	1

Таблица 2.4. Ожидание блокировки в разрезе компьютеров.

	Компьютер	Длительность	Количество
	1CV85	1ч 50м 18сек	1635

Таблица 2.5. Ожидание блокировки в разрезе пользователей.

Пользователь	Длительность	Количество
1	59м 04сек	882
2	4м 44сек	59
3	4м 07сек	69
4	4м 07сек	64
5	3м 57сек	64
6	3м 46сек	49
7	3м 23сек	51
8	2м 51сек	47
9	2м 21сек	24
10	2м 14сек	26
11	1м 54сек	40
12	1м 47сек	23
13	1м 42сек	22
14	1м 36сек	17
15	1м 28сек	32
16	1м 23сек	16
17	1м 05сек	17
18	1м 03сек	12
19	60сек	7
20	49сек	8
21	33сек	10
22	32сек	10
23	32сек	6
24	30сек	4
25	29сек	5
26	27сек	7
27	20сек	3
28	16сек	3
29	14сек	6
30	12сек	4
31	12сек	2
32	11сек	3
33	11сек	2
34	10сек	2
35	9сек	4
36	9сек	3
37	8сек	1
38	7сек	3

39	7сек	3
40	6сек	1
41	6сек	1
42	5сек	3
43	5сек	1
44	5сек	3
45	5сек	2
46	5сек	1
47	3сек	1
48	3сек	3
49	2сек	1
50	2сек	1
51	2сек	1
52	2сек	1

Таблица 2.6. Неуспешное ожидание блокировки в разрезе пользователей.

Пользователь	Длительность	Количество
1	8м 10сек	26
2	1м 20сек	4
3	1м 00сек	3
4	40сек	2
5	40сек	2
6	40сек	2
7	40сек	2
8	20сек	1
9	20сек	1
10	20сек	1
11	20сек	1

Таблица 2.7. Отмена блокировки после ожидания в разрезе компьютеров.

	Компьютер	Длительность	Количество
	1CV85	14м 30сек	45

Таблица 2.8. Ожидание блокировки в разрезе видов блокировки.

	Вид блокировки	Длительность	Количество
	Прочие	1ч 50м 18сек	1635

Таблица 2.9. Отмена блокировки после ожидания в разрезе видов блокировки.

	Вид блокировки	Длительность	Количество
	Прочие	14м 30сек	45

3. Данные за 07.11.2013 (четверг).

Таблица 3.1. Сводная таблица.

Максимальное время ожидания блокировки, мс	**686**
Среднее время ожидания блокировки, мс	208
Средняя частота ожидания блокировок в час	7,00
Общее время ожидания блокировки за время измерения	2сек
Количество ожиданий блокировки всего за время измерения	7
Общее время неуспешных ожиданий	
Количество отмен ожидания блокировки всего за время измерения	0

Таблица 3.2. Ожидание блокировки в разрезе объектов баз данных.

	Объект БД	**Длительность**	**Количество**
	8._Document4622	2сек	7

Таблица 3.4. Ожидание блокировки в разрезе компьютеров.

	Компьютер	**Длительность**	**Количество**
	1CV85	2сек	7

Таблица 3.5. Ожидание блокировки в разрезе пользователей.

Пользователь	**Длительность**	**Количество**
X	2сек	7

Таблица 3.8. Ожидание блокировки в разрезе видов блокировки.

	Вид блокировки	**Длительность**	**Количество**
	Прочие	2сек	7

4. Данные за 08.11.2013 (пятница).

Таблица 4.1. Сводная таблица.

Максимальное время ожидания блокировки, мс	20017
Среднее время ожидания блокировки, мс	3637
Средняя частота ожидания блокировок в час	495,20
Общее время ожидания блокировки за время измерения	4ч 33м 15сек
Количество ожиданий блокировки всего за время измерения	4507
Общее время неуспешных ожиданий	29м 02сек
Количество отмен ожидания блокировки всего за время измерения	92

Таблица 4.2. Ожидание блокировки в разрезе объектов баз данных.

	Объект БД	Длительность	Количество
	8.0	2ч 52м 51сек	3231
	8._BusinessProcess8775	47м 14сек	690
	8._Task4401	35м 57сек	349
	8._BusinessProcess8086	9м 06сек	88
	8._DocumentChangeRec2178	7м 20сек	126
	8._DocumentChangeRec4540	41сек	13
	8._Document49	5сек	2
	8._AccumRegChangeRec4574	2сек	4
	8._Document2042	2сек	2

Таблица 4.3. Отмена блокировки после ожидания в разрезе объектов баз данных.

	Объект БД	Длительность	Количество
	8.0	10м 22сек	36
	8._BusinessProcess8775	8м 01сек	24
	8._Task4401	6м 21сек	19
	8._BusinessProcess8086	3м 20сек	10
	8._DocumentChangeRec2178	1м 00сек	3

Таблица 4.4. Ожидание блокировки в разрезе компьютеров.

	Компьютер	Длительность	Количество
	1CV85	4ч 33м 15сек	4507

Таблица 4.5. Ожидание блокировки в разрезе пользователей.

Пользователь	Длительность	Количество
	3ч 08м 15сек	3204
1	11м 18сек	123
2	6м 58сек	123
3	6м 41сек	142
4	6м 38сек	125

5	5м 52сек	47
6	5м 10сек	69
7	4м 24сек	54
8	4м 03сек	51
9	3м 13сек	61
10	3м 09сек	49
11	3м 07сек	70
12	2м 60сек	54
13	2м 45сек	29
14	2м 31сек	30
15	2м 31сек	52
16	2м 08сек	23
17	1м 47сек	35
18	1м 35сек	16
19	1м 09сек	11
20	48сек	17
21	33сек	9
22	33сек	5
23	32сек	4
24	29сек	8
25	29сек	5
26	27сек	8
27	25сек	10
28	23сек	5
29	16сек	3
30	15сек	6
31	14сек	6
32	13сек	4
33	13сек	5
34	13сек	3
35	13сек	7
36	11сек	3
37	10сек	3
38	8сек	3
39	8сек	4
40	7сек	2
41	6сек	1
42	4сек	4
43	3сек	2
44	3сек	2
45	3сек	1
46	2сек	2
47	2сек	2

Таблица 4.6. Неуспешное ожидание блокировки в разрезе пользователей.

Пользователь	Длительность	Количество
	19м 08сек	61
1	2м 40сек	8
2	2м 00сек	6
3	41сек	2
4	41сек	2
5	40сек	2
6	40сек	2
7	24сек	2
8	21сек	1
9	21сек	1
10	20сек	1
11	20сек	1
12	20сек	1
13	20сек	1
14	11сек	1

Таблица 4.7. Отмена блокировки после ожидания в разрезе компьютеров.

	Компьютер	Длительность	Количество
	1CV85	29м 02сек	92

Таблица 4.8. Ожидание блокировки в разрезе видов блокировки.

	Вид блокировки	Длительность	Количество
	Прочие	4ч 33м 15сек	4507

Таблица 4.9. Отмена блокировки после ожидания в разрезе видов блокировки.

	Вид блокировки	Длительность	Количество
	Прочие	29м 02сек	92

5. Данные за 10.11.2013 (воскресенье).

Таблица 5.1. Сводная таблица.

Максимальное время ожидания блокировки, мс	**20000**
Среднее время ожидания блокировки, мс	20000
Средняя частота ожидания блокировок в час	2,00
Общее время ожидания блокировки за время измерения	40сек
Количество ожиданий блокировки всего за время измерения	2
Общее время неуспешных ожиданий	40сек
Количество отмен ожидания блокировки всего за время измерения	2

Таблица 5.2. Ожидание блокировки в разрезе объектов баз данных.

	Объект БД	**Длительность**	**Количество**
	8.0	40сек	2

Таблица 5.3. Отмена блокировки после ожидания в разрезе объектов баз данных.

	Объект БД	**Длительность**	**Количество**
	8.0	40сек	2

Таблица 5.4. Ожидание блокировки в разрезе компьютеров.

	Компьютер	**Длительность**	**Количество**
	1CV85	40сек	2

Таблица 5.5. Ожидание блокировки в разрезе пользователей.

	Пользователь	**Длительность**	**Количество**
	КрючковаИИ	40сек	2

Таблица 5.6. Неуспешное ожидание блокировки в разрезе пользователей.

	Пользователь	**Длительность**	**Количество**
	КрючковаИИ	40сек	2

Таблица 5.7. Отмена блокировки после ожидания в разрезе компьютеров.

	Компьютер	**Длительность**	**Количество**
	1CV85	40сек	2

Таблица 5.8. Ожидание блокировки в разрезе видов блокировки.

	Вид блокировки	Длительность	Количество
	Прочие	40сек	2

Таблица 5.9. Отмена блокировки после ожидания в разрезе видов блокировки.

	Вид блокировки	Длительность	Количество
	Прочие	40сек	2

Приложение 2. Ошибки 1С, с которыми сталкиваются пользователи

№	Ошибка	Кол-во
1	Операция не выполнена!	149
2	В данной транзакции уже происходили ошибки!	88
3	Конфликт блокировок при выполнении транзакции:_Microsoft OLE DB Provider for SQL Server: Превышено время ожидания запроса на блокировку._HRESULT=80040E31, SQLSrvr: Error state=38, Severity=10, native=1222, line=1_	72
4	Ошибка при вызове метода контекста (Записать): В данной транзакции уже происходили ошибки!	52
5	Операция не может быть выполнена из-за несоответствия версии или отсутствия записи базы данных (возможно, запись была изменена или удалена)!	49
6	Ошибка при вызове метода контекста (ПолучитьОбъект): В данной транзакции уже происходили ошибки!	39
7	Ошибка при получении значения атрибута контекста (БумКопия): В данной транзакции уже происходили ошибки!	35
8	Ошибка при вызове метода контекста (Записать): Ошибка при выполнении обработчика - _Ошибка при вызове метода контекста (ПолучитьОбъект): В данной транзакции уже происходили ошибки!_	14
9	Ошибка при вызове метода контекста (Записать): Ошибка совместного доступа к файлу _C:\Documents and Settings\AvdoninaOV\Local Settings\Temp\.tif_	6
10	Не удалось заблокировать запись. Действие (изменение, удаление или блокировка записи) не выполнено._Запись заблокирована пользователем _, с компьютера _WS356_, из приложения _1С:Предприятие_, соединение _727_.	5
11	Ошибка при вызове метода контекста (Записать): Ошибка при выполнении обработчика - _Ошибка при вызове метода контекста (Записать): Ошибка при выполнении обработчика - _{Документ.КарточкаДоговора(1491)}: Ошибка при вызове метода контекста (Выполнить): Ошиб	5
12	Ошибка при вызове метода контекста (Записать): Ошибка совместного доступа к файлу _C:\Documents and Settings\KiselevaOA\Local Settings\Temp\Маурер.tif_	5
13	Ошибка при получении значения атрибута контекста (ОП): В данной транзакции уже происходили ошибки!	5
14	Поле объекта не обнаружено (ДействиеДокументыВыданы_ВозвратПодписанногоДоговора)	5
15	Ошибка при вызове метода контекста (Записать): Ошибка совместного доступа к файлу _C:\Documents and Settings\Local Settings\Temp\905916.tif_	4
16	Не удалось заблокировать запись. Действие (изменение, удаление или блокировка записи) не выполнено._Запись заблокирована пользователем _, с компьютера _WS345_, из приложения _1С:Предприятие_, соединение _1388_.	3
17	Ошибка при вызове метода контекста (Выполнить): Ошибка выполнения запроса "Конфликт блокировок при выполнении транзакции:_Microsoft OLE DB Provider for SQL Server: Превышено время ожидания запроса на блокировку._HRESULT=80040E31, SQLSrvr: Error state=34,	3
18	Ошибка при вызове метода контекста (Записать): Конфликт блокировок при выполнении транзакции:_Microsoft OLE DB Provider for SQL Server: Транзакция (идентификатор процесса 112) вызвала взаимоблокировку ресурсов блокировка с другим процессом и стала жертвой	3
19	Ошибка при вызове метода контекста (Записать): Ошибка при выполнении обработчика - _Ошибка при вызове метода контекста (Выполнить): Ошибка выполнения запроса "Конфликт блокировок при выполнении транзакции:_Microsoft OLE DB Provider for SQL Server: Транзак	3
20	{(134, 39)}: Не задано значение параметра "Иски"_ИЛИ КарточкаДоговора.Статус = <>&Иски	2

Приложение 3. Скрипт переиндексации таблиц базы данных

```
SET NOCOUNT ON
DECLARE       @cmd  varchar(8000), @sqlName    varchar(256), @1cName      varchar(256),
@ErrCode      int

IF object_id('rs_dbreindex_result') is null
       CREATE TABLE [rs_dbreindex_result]
              (id    int identity (1,1)
              ,_sqlName     varchar(128)
              ,_1cName      varchar(256)
              ,ErrCode      int
              ,Date         datetime)

TRUNCATE TABLE rs_dbreindex_result
DECLARE cur1 CURSOR FOR

       SELECT a.[name], a.[name]
       FROM sysobjects a (nolock)
       WHERE a.xtype = 'U'
       ORDER BY a.[name]
OPEN cur1
FETCH NEXT FROM cur1 INTO @sqlName,@1cName

WHILE @@FETCH_STATUS = 0
BEGIN

       SET @cmd = 'dbcc dbreindex (''' + @sqlName + ''')'
       begin tran
              EXEC(@cmd)
              SET @ErrCode = @@ERROR
       if @ErrCode<> 0
       begin
              PRINT 'Ошибка при пересоздания индексов на ' + @sqlname      + ' код ошибки:
'+ CAST( @ErrCode AS varchar)
              ROLLBACK
       end
       ELSE
       begin
              print @cmd + '    --- Успешно'
              commit
       end

      INSERT rs_dbreindex_result SELECT @sqlName, @1cName, @ErrCode, getdate()

       /*
       IF @ErrCode <> 0
              PRINT 'Ошибка при пересоздания индексов на ' + @sqlname
       ELSE
              PRINT @sqlname + ' Успешно'
       */

       -- print @cmd

       FETCH NEXT FROM cur1 INTO @sqlName,@1cName
END
CLOSE cur1
DEALLOCATE cur1
-- select * from rs_dbreindex_result
```

Приложение 4. Список запросов, создающих наибольшую нагрузку на процессор

Статистика собрана за период с 4 по 11 ноября 2013г.

№	База данных №	Запрос	Кол. запросов	сумма CPU	среднее CPU	%доля CPU	макс. CPU	%доля чтение
1	8	exec sp_executesql N'SELECT _Task4401_Q_000_T_001._IDRRef AS _Q_000_F_000RRef FROM _Task4401 _Task4401_Q_000_T_001 WITH(NOLOCK) LEFT OUTER JOIN _BusinessProcess8775 WITH(NOLOCK) ON _Task4401_Q_000_T_001._BusinessProcess_TYPE = 0x08 AND _Task4401_Q_000_T_0	536	1ч 40м	11,26с	17,38 %	16,84с	1,51%
2	8	exec sp_executesql N'SELECT _Task4401_Q_000_T_001._IDRRef AS _Q_000_F_000RRef FROM _Task4401 _Task4401_Q_000_T_001 WITH(NOLOCK) LEFT OUTER JOIN _BusinessProcess8775 WITH(NOLOCK) ON _Task4401_Q_000_T_001._BusinessProcess_TYPE = 0x08 AND _Task4401_Q_000_T_0	310	58м 55с	11,41с	10,19 %	15,75с	0,87%
3	8	exec sp_executesql N'SELECT DISTINCT CAST(NULL AS BINARY(1)) AS f_1, #V8TblAli1_Q_000_T_001._Q_004_F_018 AS f_2, #V8TblAli1_Q_000_T_001._Q_004_F_020 AS f_3, #V8TblAli1_Q_000_T_001._Q_004_F_005 AS f_4, #V8TblAli1_Q_000_T_001._Q_004_F_006 AS f_5, #V8TblAli1	101	56м 21с	33,48с	9,74%	1м 2с	6,33%
4	8	exec sp_executesql N'SELECT DISTINCT CAST(NULL AS BINARY(1)) AS f_1, #V8TblAli1_Q_000_T_001._Q_004_F_018 AS f_2, #V8TblAli1_Q_000_T_001._Q_004_F_020 AS f_3, #V8TblAli1_Q_000_T_001._Q_004_F_005 AS f_4, #V8TblAli1_Q_000_T_001._Q_004_F_006 AS f_5, #V8TblAli1	89	52м 42с	35,53с	9,11%	57,46с	6,69%
5	8	exec sp_executesql N'SELECT DISTINCT CAST(NULL AS BINARY(1)) AS f_1, #V8TblAli1_Q_000_T_001._Q_004_F_018 AS f_2, #V8TblAli1_Q_000_T_001._Q_004_F_020 AS f_3, #V8TblAli1_Q_000_T_001._Q_004_F_005 AS f_4, #V8TblAli1_Q_000_T_001._Q_004_F_006 AS f_5, #V8TblAli1	103	49м 14с	28,68с	8,51%	51,82с	5,14%
6	8	exec sp_executesql N'SELECT _Task4401_Q_000_T_001._BusinessProcess_TYPE AS f_1, _Task4401_Q_000_T_001._BusinessProcess_RTRef AS f_2, _Task4401_Q_000_T_001._BusinessProcess_RRRef AS f_3 FROM _Task4401 _Task4401_Q_000_T_001 WITH(REPEATABLEREAD) LEFT OUTER J	212	23м 6с	6,54с	3,99%	9,72с	1,51%
7	8	exec sp_executesql N'SELECT #V8TblAli1_Q_000_T_001._Fld3091_TYPE AS f_1, #V8TblAli1_Q_000_T_001._Fld3091_RTRef AS f_2, #V8TblAli1_Q_000_T_001._Fld3091_RRRef AS f_3, #V8TblAli1_Q_000_T_001._Fld3092_TYPE AS f_4, #V8TblAli1_Q_000_T_001._Fld3092_RTRef AS f_5,	19	20м 32с	1м 4с	3,55%	1м 10с	15,36%
8	8	exec sp_executesql N'SELECT DISTINCT CAST(NULL AS BINARY(1)) AS f_1, #V8TblAli1_Q_000_T_001._Q_004_F_018 AS f_2, #V8TblAli1_Q_000_T_001._Q_004_F_020 AS f_3, #V8TblAli1_Q_000_T_001._Q_004_F_005 AS f_4, #V8TblAli1_Q_000_T_001._Q_004_F_006 AS f_5, #V8TblAli1	28	20м 12с	43,30с	3,49%	53,12с	2,66%
9	8	exec sp_executesql N'SELECT #V8TblAli1_Q_000_T_001._Fld3091_TYPE AS f_1, #V8TblAli1_Q_000_T_001._Fld3091_RTRef AS f_2, #V8TblAli1_Q_000_T_001._Fld3091_RRRef AS f_3, #V8TblAli1_Q_000_T_001._Fld3092_TYPE AS f_4, #V8TblAli1_Q_000_T_001._Fld3092_RTRef AS f_5,	16	17м 1с	1м 3с	2,94%	1м 12с	12,92%
10	8	exec sp_executesql N'SELECT _Task4401_Q_000_T_001._IDRRef AS _Q_000_F_000RRef FROM _Task4401 _Task4401_Q_000_T_001 WITH(NOLOCK) LEFT OUTER JOIN _BusinessProcess8775 WITH(NOLOCK) ON _Task4401_Q_000_T_001._BusinessProcess_TYPE = 0x08 AND _Task4401_Q_000_T_0	80	15м 36с	11,70с	2,70%	15,88с	0,23%
11	8	exec sp_executesql N'SELECT _Task4401_Q_000_T_001._IDRRef AS f_1, _Task4401_Q_000_T_001._BusinessProcess_TYPE AS f_2, _Task4401_Q_000_T_001._BusinessProcess_RTRef AS f_3, _Task4401_Q_000_T_001._BusinessProcess_RRRef AS f_4 FROM _Task4401 _Task4401_Q_000_T	92	14м 11с	9,26с	2,45%	12,25с	0,68%

12	8	exec sp_executesql N'SELECT #V8TblAli1_Q_000_T_001._Q_009_F_000 AS f_1, #V8TblAli1_Q_000_T_001._Q_009_F_001 AS f_2, #V8TblAli1_Q_000_T_001._Q_009_F_002 AS f_3, #V8TblAli1_Q_000_T_001._Q_009_F_003 AS f_4, #V8TblAli1_Q_000_T_001._Q_009_F_004 AS f_5, #V8TblA	14	12м 15с	52,54с	2,12%	1м 42с	1,18%
13	8	exec sp_executesql N'SELECT DISTINCT CAST(NULL AS BINARY(1)) AS f_1, #V8TblAli1_Q_000_T_001._Q_004_F_018 AS f_2, #V8TblAli1_Q_000_T_001._Q_004_F_020 AS f_3, #V8TblAli1_Q_000_T_001._Q_004_F_005 AS f_4, #V8TblAli1_Q_000_T_001._Q_004_F_006 AS f_5, #V8TblAli1	16	9м 17с	34,82с	1,60%	58,70с	0,95%
14	8	exec sp_executesql N'SELECT _Task4401_Q_000_T_001._IDRRef AS f_1, _Task4401_Q_000_T_001._BusinessProcess_TYPE AS f_2, _Task4401_Q_000_T_001._BusinessProcess_RTRef AS f_3, _Task4401_Q_000_T_001._BusinessProcess_RRRef AS f_4 FROM _Task4401 _Task4401_Q_000_T	57	8м 50с	9,30с	1,53%	12,36с	0,42%
15	8	exec sp_executesql N'SELECT #V8TblAli1_Q_000_T_001._Fld3091_TYPE AS f_1, #V8TblAli1_Q_000_T_001._Fld3091_RTRef AS f_2, #V8TblAli1_Q_000_T_001._Fld3091_RRRef AS f_3, #V8TblAli1_Q_000_T_001._Fld3092_TYPE AS f_4, #V8TblAli1_Q_000_T_001._Fld3092_RTRef AS f_5,	7	7м 55с	1м 7с	1,37%	1м 11с	5,65%
16	8	exec sp_executesql N'INSERT INTO #tt3 (_Q_000_F_000_TYPE, _Q_000_F_000_RTRef, _Q_000_F_000_RRRef, _Q_000_F_001) SELECT _InfoReg6558_Q_000_T_001._Fld6559_TYPE AS _Q_000_F_000_TYPE, _InfoReg6558_Q_000_T_001._Fld6559_RTRef AS _Q_000_F_000_RTRef, _InfoReg6558	608	7м 41с	0,76с	1,33%	1,41с	1,02%
17	8	exec sp_executesql N'SELECT DISTINCT CAST(NULL AS BINARY(1)) AS f_1, #V8TblAli1_Q_000_T_001._Q_004_F_018 AS f_2, #V8TblAli1_Q_000_T_001._Q_004_F_020 AS f_3, #V8TblAli1_Q_000_T_001._Q_004_F_005 AS f_4, #V8TblAli1_Q_000_T_001._Q_004_F_006 AS f_5, #V8TblAli1	14	7м 38с	32,78с	1,32%	39,92с	0,19%
18	8	exec sp_executesql N'SELECT _Task4401_Q_000_T_001._BusinessProcess_TYPE AS f_1, _Task4401_Q_000_T_001._BusinessProcess_RTRef AS f_2, _Task4401_Q_000_T_001._BusinessProcess_RRRef AS f_3 FROM _Task4401 _Task4401_Q_000_T_001 WITH(REPEATABLEREAD) LEFT OUTER J	63	6м 3с	5,77с	1,05%	8,19с	0,09%
19	8	exec sp_executesql N'SELECT DISTINCT _Task4401_Q_000_T_001._BusinessProcess_TYPE AS f_1, _Task4401_Q_000_T_001._BusinessProcess_RTRef AS f_2, _Task4401_Q_000_T_001._BusinessProcess_RRRef AS f_3, _Task4401_Q_000_T_001._IDRRef AS f_4 FROM _Task4401 _Task440	60	5м 41с	5,69с	0,98%	7,63с	0,43%
20	8	exec sp_executesql N'SELECT DISTINCT CAST(NULL AS BINARY(1)) AS f_1, #V8TblAli1_Q_000_T_001._Q_004_F_018 AS f_2, #V8TblAli1_Q_000_T_001._Q_004_F_020 AS f_3, #V8TblAli1_Q_000_T_001._Q_004_F_005 AS f_4, #V8TblAli1_Q_000_T_001._Q_004_F_006 AS f_5, #V8TblAli1	8	5м 29с	41,14с	0,95%	53,25с	0,81%
21	8	exec sp_executesql N'SELECT #V8TblAli1_Q_000_T_001._Fld5032_TYPE AS _sf_1_TYPE, #V8TblAli1_Q_000_T_001._Fld5032_RTRef AS _sf_1_RTRef, #V8TblAli1_Q_000_T_001._Fld5032_RRRef AS _sf_1_RRRef, CASE WHEN #V8TblAli2_Q_000_T_002._Fld5042_TYPE = 0x08 AND #V8TblAli	1	5м 14с	5м 14с	0,91%	5м 14с	2,50%
22	8	exec sp_executesql N'SELECT #V8TblAli1_Q_000_T_001._Fld5032_TYPE AS _sf_1_TYPE, #V8TblAli1_Q_000_T_001._Fld5032_RTRef AS _sf_1_RTRef, #V8TblAli1_Q_000_T_001._Fld5032_RRRef AS _sf_1_RRRef, CASE WHEN #V8TblAli2_Q_000_T_002._Fld5042_TYPE = 0x08 AND #V8TblAli	1	5м 7с	5м 7с	0,89%	5м 7с	2,50%

Приложение 5. Запрос, создающий наибольшую нагрузку на процессор

```
exec sp_executesql N'
SELECT _Task4401_Q_000_T_001._IDRRef AS _Q_000_F_000RRef
FROM _Task4401 _Task4401_Q_000_T_001 WITH (NOLOCK)
LEFT OUTER JOIN _BusinessProcess8775 WITH (NOLOCK) ON
_Task4401_Q_000_T_001._BusinessProcess_TYPE = 0x08
       AND _Task4401_Q_000_T_001._BusinessProcess_RTRef = 0x00002247
       AND _Task4401_Q_000_T_001._BusinessProcess_RRRef = _BusinessProcess8775._IDRRef
LEFT OUTER JOIN _BusinessProcess9197 WITH (NOLOCK) ON
_Task4401_Q_000_T_001._BusinessProcess_TYPE = 0x08
       AND _Task4401_Q_000_T_001._BusinessProcess_RTRef = 0x000023ED
       AND _Task4401_Q_000_T_001._BusinessProcess_RRRef = _BusinessProcess9197._IDRRef
LEFT OUTER JOIN _BusinessProcess4399 WITH (NOLOCK) ON
_Task4401_Q_000_T_001._BusinessProcess_TYPE = 0x08
       AND _Task4401_Q_000_T_001._BusinessProcess_RTRef = 0x0000112F
       AND _Task4401_Q_000_T_001._BusinessProcess_RRRef = _BusinessProcess4399._IDRRef
LEFT OUTER JOIN _BusinessProcess6191 WITH (NOLOCK) ON
_Task4401_Q_000_T_001._BusinessProcess_TYPE = 0x08
       AND _Task4401_Q_000_T_001._BusinessProcess_RTRef = 0x0000182F
       AND _Task4401_Q_000_T_001._BusinessProcess_RRRef = _BusinessProcess6191._IDRRef
LEFT OUTER JOIN _BusinessProcess8086 WITH (NOLOCK) ON
_Task4401_Q_000_T_001._BusinessProcess_TYPE = 0x08
       AND _Task4401_Q_000_T_001._BusinessProcess_RTRef = 0x00001F96
       AND _Task4401_Q_000_T_001._BusinessProcess_RRRef = _BusinessProcess8086._IDRRef
LEFT OUTER JOIN _BusinessProcess4956 WITH (NOLOCK) ON
_Task4401_Q_000_T_001._BusinessProcess_TYPE = 0x08
       AND _Task4401_Q_000_T_001._BusinessProcess_RTRef = 0x0000135C
       AND _Task4401_Q_000_T_001._BusinessProcess_RRRef = _BusinessProcess4956._IDRRef
WHERE CASE
              WHEN _Task4401_Q_000_T_001._BusinessProcess_TYPE = 0x08
                     AND _Task4401_Q_000_T_001._BusinessProcess_RTRef = 0x0000135C
                     THEN _BusinessProcess4956._Fld4962RRef
              WHEN _Task4401_Q_000_T_001._BusinessProcess_TYPE = 0x08
                     AND _Task4401_Q_000_T_001._BusinessProcess_RTRef = 0x00001F96
                     THEN _BusinessProcess8086._Fld8099RRef
              WHEN _Task4401_Q_000_T_001._BusinessProcess_TYPE = 0x08
                     AND _Task4401_Q_000_T_001._BusinessProcess_RTRef = 0x0000182F
                     THEN _BusinessProcess6191._Fld6220RRef
              WHEN _Task4401_Q_000_T_001._BusinessProcess_TYPE = 0x08
                     AND _Task4401_Q_000_T_001._BusinessProcess_RTRef = 0x0000112F
                     THEN _BusinessProcess4399._Fld4465RRef
              WHEN _Task4401_Q_000_T_001._BusinessProcess_TYPE = 0x08
                     AND _Task4401_Q_000_T_001._BusinessProcess_RTRef = 0x000023ED
                     THEN _BusinessProcess9197._Fld9216RRef
              WHEN _Task4401_Q_000_T_001._BusinessProcess_TYPE = 0x08
                     AND _Task4401_Q_000_T_001._BusinessProcess_RTRef = 0x00002247
                     THEN _BusinessProcess8775._Fld8778RRef
              ELSE CAST(NULL AS BINARY (16))
              END = @P1
       AND CASE
              WHEN _Task4401_Q_000_T_001._BusinessProcess_TYPE = 0x08
                     AND _Task4401_Q_000_T_001._BusinessProcess_RTRef = 0x0000135C
                     THEN _BusinessProcess4956._Started
              WHEN _Task4401_Q_000_T_001._BusinessProcess_TYPE = 0x08
                     AND _Task4401_Q_000_T_001._BusinessProcess_RTRef = 0x00001F96
                     THEN _BusinessProcess8086._Started
              WHEN _Task4401_Q_000_T_001._BusinessProcess_TYPE = 0x08
                     AND _Task4401_Q_000_T_001._BusinessProcess_RTRef = 0x0000182F
                     THEN _BusinessProcess6191._Started
```

```
                WHEN _Task4401_Q_000_T_001._BusinessProcess_TYPE = 0x08
                      AND _Task4401_Q_000_T_001._BusinessProcess_RTRef = 0x0000112F
                      THEN _BusinessProcess4399._Started
                WHEN _Task4401_Q_000_T_001._BusinessProcess_TYPE = 0x08
                      AND _Task4401_Q_000_T_001._BusinessProcess_RTRef = 0x000023ED
                      THEN _BusinessProcess9197._Started
                WHEN _Task4401_Q_000_T_001._BusinessProcess_TYPE = 0x08
                      AND _Task4401_Q_000_T_001._BusinessProcess_RTRef = 0x00002247
                      THEN _BusinessProcess8775._Started
                ELSE CAST(NULL AS BINARY (1))
                END = @P2

UNION ALL

SELECT _Task4401_Q_001_T_001._IDRRef AS _Q_000_F_000RRef
FROM _Task4401 _Task4401_Q_001_T_001 WITH (NOLOCK)
LEFT OUTER JOIN _BusinessProcess8775 WITH (NOLOCK) ON
_Task4401_Q_001_T_001._BusinessProcess_TYPE = 0x08
      AND _Task4401_Q_001_T_001._BusinessProcess_RTRef = 0x00002247
      AND _Task4401_Q_001_T_001._BusinessProcess_RRRef = _BusinessProcess8775._IDRRef
LEFT OUTER JOIN _BusinessProcess9197 WITH (NOLOCK) ON
_Task4401_Q_001_T_001._BusinessProcess_TYPE = 0x08
      AND _Task4401_Q_001_T_001._BusinessProcess_RTRef = 0x000023ED
      AND _Task4401_Q_001_T_001._BusinessProcess_RRRef = _BusinessProcess9197._IDRRef
LEFT OUTER JOIN _BusinessProcess4399 WITH (NOLOCK) ON
_Task4401_Q_001_T_001._BusinessProcess_TYPE = 0x08
      AND _Task4401_Q_001_T_001._BusinessProcess_RTRef = 0x0000112F
      AND _Task4401_Q_001_T_001._BusinessProcess_RRRef = _BusinessProcess4399._IDRRef
LEFT OUTER JOIN _BusinessProcess6191 WITH (NOLOCK) ON
_Task4401_Q_001_T_001._BusinessProcess_TYPE = 0x08
      AND _Task4401_Q_001_T_001._BusinessProcess_RTRef = 0x0000182F
      AND _Task4401_Q_001_T_001._BusinessProcess_RRRef = _BusinessProcess6191._IDRRef
LEFT OUTER JOIN _BusinessProcess8086 WITH (NOLOCK) ON
_Task4401_Q_001_T_001._BusinessProcess_TYPE = 0x08
      AND _Task4401_Q_001_T_001._BusinessProcess_RTRef = 0x00001F96
      AND _Task4401_Q_001_T_001._BusinessProcess_RRRef = _BusinessProcess8086._IDRRef
LEFT OUTER JOIN _BusinessProcess4956 WITH (NOLOCK) ON
_Task4401_Q_001_T_001._BusinessProcess_TYPE = 0x08
      AND _Task4401_Q_001_T_001._BusinessProcess_RTRef = 0x0000135C
      AND _Task4401_Q_001_T_001._BusinessProcess_RRRef = _BusinessProcess4956._IDRRef
WHERE CASE
                WHEN _Task4401_Q_001_T_001._BusinessProcess_TYPE = 0x08
                      AND _Task4401_Q_001_T_001._BusinessProcess_RTRef = 0x0000135C
                      THEN _BusinessProcess4956._Completed
                WHEN _Task4401_Q_001_T_001._BusinessProcess_TYPE = 0x08
                      AND _Task4401_Q_001_T_001._BusinessProcess_RTRef = 0x00001F96
                      THEN _BusinessProcess8086._Completed
                WHEN _Task4401_Q_001_T_001._BusinessProcess_TYPE = 0x08
                      AND _Task4401_Q_001_T_001._BusinessProcess_RTRef = 0x0000182F
                      THEN _BusinessProcess6191._Completed
                WHEN _Task4401_Q_001_T_001._BusinessProcess_TYPE = 0x08
                      AND _Task4401_Q_001_T_001._BusinessProcess_RTRef = 0x0000112F
                      THEN _BusinessProcess4399._Completed
                WHEN _Task4401_Q_001_T_001._BusinessProcess_TYPE = 0x08
                      AND _Task4401_Q_001_T_001._BusinessProcess_RTRef = 0x000023ED
                      THEN _BusinessProcess9197._Completed
                WHEN _Task4401_Q_001_T_001._BusinessProcess_TYPE = 0x08
                      AND _Task4401_Q_001_T_001._BusinessProcess_RTRef = 0x00002247
                      THEN _BusinessProcess8775._Completed
                ELSE CAST(NULL AS BINARY (1))
                END = @P2
      AND CASE
                WHEN _Task4401_Q_001_T_001._BusinessProcess_TYPE = 0x08
                      AND _Task4401_Q_001_T_001._BusinessProcess_RTRef = 0x0000135C
```

```
            THEN _BusinessProcess4956._Fld4962RRef
       WHEN _Task4401_Q_001_T_001._BusinessProcess_TYPE = 0x08
            AND _Task4401_Q_001_T_001._BusinessProcess_RTRef = 0x00001F96
            THEN _BusinessProcess8086._Fld8099RRef
       WHEN _Task4401_Q_001_T_001._BusinessProcess_TYPE = 0x08
            AND _Task4401_Q_001_T_001._BusinessProcess_RTRef = 0x0000182F
            THEN _BusinessProcess6191._Fld6220RRef
       WHEN _Task4401_Q_001_T_001._BusinessProcess_TYPE = 0x08
            AND _Task4401_Q_001_T_001._BusinessProcess_RTRef = 0x0000112F
            THEN _BusinessProcess4399._Fld4465RRef
       WHEN _Task4401_Q_001_T_001._BusinessProcess_TYPE = 0x08
            AND _Task4401_Q_001_T_001._BusinessProcess_RTRef = 0x000023ED
            THEN _BusinessProcess9197._Fld9216RRef
       WHEN _Task4401_Q_001_T_001._BusinessProcess_TYPE = 0x08
            AND _Task4401_Q_001_T_001._BusinessProcess_RTRef = 0x00002247
            THEN _BusinessProcess8775._Fld8778RRef
       ELSE CAST(NULL AS BINARY (16))
       END = @P1 ', N' @P1 VARBINARY(16)
,@P2 VARBINARY(1) ', 0x902B984BE10C268C11E2A8AD5B09668C, 0x01'
```

Приложение 6. Список запросов, вызывающих наибольшее количество логических чтений

Статистика собрана за период с 4 по 11 ноября 2013г.

№	Запрос	Кол. запросов	%доля CPU	сумма чтение	среднее чтение	%доля чтение	макс. чтение
1	exec sp_executesql N'SELECT _Reference2_Q_000_T_001._IDRRef AS f_1 FROM _Reference2 _Reference2_Q_000_T_001 WITH(NOLOCK) WHERE _Reference2_Q_000_T_001._Fld116RRef = @P1 AND _Reference2_Q_000_T_001._Fld115_TYPE = @P2 AND _Reference2_Q_000_T_001._Fld115_S =	113 089	4,49%	9 379 346 432	82 937	34,15%	89 385
2	exec sp_executesql N'SELECT _Task4401_Q_000_T_001._IDRRef AS f_1 FROM _Task4401 _Task4401_Q_000_T_001 WITH(NOLOCK) LEFT OUTER JOIN _BusinessProcess8775 WITH(NOLOCK) ON _Task4401_Q_000_T_001._BusinessProcess_TYPE = 0x08 AND _Task4401_Q_000_T_001._BusinessP	70 189	21,87 %	5 675 682 816	80 862	20,67%	81 701
3	exec sp_executesql N'SELECT DISTINCT _Task4401_Q_000_T_001._BusinessProcess_TYPE AS f_1, _Task4401_Q_000_T_001._BusinessProcess_RTRef AS f_2, _Task4401_Q_000_T_001._BusinessProcess_RRRef AS f_3, _Task4401_Q_000_T_001._IDRRef AS f_4 FROM _Task4401 _Task440	13 353	4,23%	1 080 562 176	80 922	3,93%	82 259
4	exec sp_executesql N'SELECT _Task4401_Q_000_T_001._Point_TYPE AS f_1, _Task4401_Q_000_T_001._Point_RTRef AS f_2, _Task4401_Q_000_T_001._Point_RRRef AS f_3, CASE WHEN _Task4401_Q_000_T_001._BusinessProcess_TYPE = 0x08 AND _Task4401_Q_000_T_001._BusinessPro	11 714	3,67%	946 139 008	80 769	3,45%	81 667
5	SELECT TOP 42 _Document2042_R._Date_Time AS _A1, _Document2042_R._Number AS _A2, _Document2042_R._Fld2094RRef AS _A3RRef, _Document2042_R._Fld2182 AS _A4, _Document2042_R._Fld2110RRef AS _A5RRef, _Document2042_R._Fld2097 AS _A6, _Document2042_R._Fld2100 A	1 083	0,43%	922 310 016	851 625	3,36%	853 137
6	exec sp_executesql N'SELECT DISTINCT _Task4401_Q_000_T_001._BusinessProcess_TYPE AS f_1, _Task4401_Q_000_T_001._BusinessProcess_RTRef AS f_2, _Task4401_Q_000_T_001._BusinessProcess_RRRef AS f_3, _Task4401_Q_000_T_001._IDRRef AS f_4 FROM _Task4401 _Task440	10 787	14,47 %	863 101 056	80 013	3,14%	80 949
7	exec sp_executesql N'SELECT _Task4401_Q_000_T_001._IDRRef AS f_1, _Task4401_Q_000_T_001._Point_TYPE AS f_2, _Task4401_Q_000_T_001._Point_RTRef AS f_3, _Task4401_Q_000_T_001._Point_RRRef AS f_4, _Task4401_Q_000_T_001._BusinessProcess_TYPE AS f_5, _Task4401	7 877	2,71%	637 176 832	80 890	2,32%	81 701
8	exec sp_executesql N'SELECT TOP 42 _Document2042_R._Date_Time AS _A1, _Document2042_R._Number AS _A2, _Document2042_R._Fld2094RRef AS _A3RRef, _Document2042_R._Fld2182 AS _A4, _Document2042_R._Fld2110RRef AS _A5RRef, _Document2042_R._Fld2097 AS _A6, _Docu	659	0,25%	551 893 056	837 470	2,01%	852 727
9	exec sp_executesql N'SELECT _Task4401_Q_000_T_001._IDRRef AS f_1 FROM _Task4401 _Task4401_Q_000_T_001 WITH(REPEATABLEREAD) LEFT OUTER JOIN _BusinessProcess8775 WITH(REPEATABLEREAD) ON _Task4401_Q_000_T_001._BusinessProcess_TYPE = 0x08 AND _Task4401_Q_000_	6 425	2,47%	520 193 952	80 964	1,89%	82 544

10	exec sp_executesql N'SELECT _Task4401_Q_000_T_001._BusinessProcess_TYPE AS f_1, _Task4401_Q_000_T_001._BusinessProcess_RTRef AS f_2, _Task4401_Q_000_T_001._BusinessProcess_RRRef AS f_3 FROM _Task4401 _Task4401_Q_000_T_001 WITH(REPEATABLEREAD) LEFT OUTER J	6 198	9,02%	505 155 232	81 502	1,84%	82 465
11	exec sp_executesql N'SELECT TOP 46 _Document2042_R._Date_Time AS _A1, _Document2042_R._Number AS _A2, _Document2042_R._Fld2094RRef AS _A3RRef, _Document2042_R._Fld2182 AS _A4, _Document2042_R._Fld2110RRef AS _A5RRef, _Document2042_R._Fld2097 AS _A6, _Docu	517	0,16%	439 954 176	850 975	1,60%	853 000
12	exec sp_executesql N'SELECT _Task4401_Q_000_T_001._IDRRef AS f_1, _Task4401_Q_000_T_001._BusinessProcess_TYPE AS f_2, _Task4401_Q_000_T_001._BusinessProcess_RTRef AS f_3, _Task4401_Q_000_T_001._BusinessProcess_RRRef AS f_4 FROM _Task4401 _Task4401_Q_000_T	5 046	10,77 %	417 244 192	82 688	1,52%	83 529
13	exec sp_executesql N'SELECT DISTINCT _Task4401_Q_000_T_001._BusinessProcess_TYPE AS f_1, _Task4401_Q_000_T_001._BusinessProcess_RTRef AS f_2, _Task4401_Q_000_T_001._BusinessProcess_RRRef AS f_3, _Task4401_Q_000_T_001._IDRRef AS f_4 FROM _Task4401 _Task440	4 863	1,61%	393 411 392	80 898	1,43%	81 696
14	exec sp_executesql N'SELECT _Task4401_Q_000_T_001._IDRRef AS f_1 FROM _Task4401 _Task4401_Q_000_T_001 WITH(NOLOCK) LEFT OUTER JOIN _BusinessProcess8775 WITH(NOLOCK) ON _Task4401_Q_000_T_001._BusinessProcess_TYPE = 0x08 AND _Task4401_Q_000_T_001._BusinessP	4 688	1,50%	379 237 280	80 895	1,38%	81 274
15	exec sp_executesql N'SELECT _Task4401_Q_000_T_001._IDRRef AS f_1, _Task4401_Q_000_T_001._Point_TYPE AS f_2, _Task4401_Q_000_T_001._Point_RTRef AS f_3, _Task4401_Q_000_T_001._Point_RRRef AS f_4, _Task4401_Q_000_T_001._BusinessProcess_TYPE AS f_5, _Task4401	4 374	1,44%	342 593 792	78 325	1,25%	79 044

Приложение 7. Пример запроса, создающего большое количество логических чтений

```
exec sp_executesql N'
SELECT
_Reference2_Q_000_T_001._IDRRef AS f_1
FROM
_Reference2 _Reference2_Q_000_T_001 WITH(NOLOCK)
WHERE
_Reference2_Q_000_T_001._Fld116RRef = @P1
       AND _Reference2_Q_000_T_001._Fld115_TYPE = @P2
       AND _Reference2_Q_000_T_001._Fld115_S = @P3
       AND _Reference2_Q_000_T_001._Fld117
              LIKE N'' У вас на согласовании находится документ.
(указан в поле "Сопроводительный документ")

Окончание срока согласования: 19.02.2010 0:00:00
Дополнительное соглашение 082-000358 от 25.03.2008 11:54:05''',N'@P1 varbinary(16),@P2
varbinary(1),@P3 nvarchar(31)',0x972D001A4BF1295C11DEC5E364E56853,0x05,N'Система
согласования документов'
```