Механизм транзакций

Статья на ИТС

Транзакция – это последовательность действий, переводящая базу данных из одного целостного состояния в другое целостное состояние.

Свойства транзакции:

- Атомарность (неделимость). После завершения транзакции все данные должны быть согласованы. Даже при простом добавлении записи может произойти рассогласование, и необходимо это обрабатывать.
- Изоляция. Это свойство обеспечивает параллельную работу пользователей и предотвращает порчу общих данных. Например, чтобы не вышло ситуации, когда 2 пользователя меняют один и тот же документ и тем самым перестирают данные друг друга. Блокировки выступают как средство обеспечения изоляции транзакции.

Особенности транзакции:

- не поддерживаются вложенные транзакции. Так эта особенность официально звучит, однако интересно: далее в тексте встречается термин Вложенная транзакция. Понять и простить. Под этим подразумевается, что вместо дерева транзакций мы имеем дело с одноуровневым списком, и все должно быть выполнено. В случае возникновения и подавлении ошибки где-то внутри "вложенной" транзакции, все равно вся транзакция будет считаться битой.

Ситуация в коде	Ожидание	Реальность
Начало транзакции 1 Вызов функции А Начало транзакции 2 Конец транзакции 2 Вызов функции Б Начало транзакции 3 Ошибка, обработка исключения Конец транзакции 3 Конец транзакции 1	Начало транзакции 1 Начало транзакции 2 Конец транзакции 2 Начало транзакции 3 Ошибка, обработка исключения Конец транзакции 3 Конец транзакции 1 Транзакция 1 должна завершиться правильно, т к ошибка в транзакции 3 обработана	Начало транзакции 1 Начало транзакции 2 Конец транзакции 2 Начало транзакции 3 Ошибка, обработка исключения Конец транзакции 3 Конец транзакции 1 Транзакция 1 завершится возвратом в любом случае. А если транзакция 3 еще и не отправит ошибку вверх по стеку, (пример кода ниже), то при разборе полетов будет ничего не понятно.

  Здесь помогает метод ТранзакцияАктивна, помогающий текущему коду понять, находится ли он внутри транзакции или его транзакция наверху.
- при возникновении исключения в общем случае транзакция не может быть зафиксирована – при этом не важно, было ли это исключение обработано или нет
- транзакция может быть инициирована явно в прикладном коде при использовании метода НачатьТранзакцию. Так же платформа неявным образом начинает транзакцию при любой записи в базу данных //Добавлено после экспериментов//

Некоторые события (например ПриЗаписиНаСервере) входят в состав транзакции, автоматически созданной платформой. Поэтому в некоторых источниках вся процедура обработки данного события входит в транзакцию (причем код, начинающий и заканчивающий транзакцию где-то посередине), что приводит к неправильным выводам (в моем случае - "то есть если в коде присутствует начало и конец транзакции, то вся процедура попадает в транзакцию", "а зачем тогда посередине начинать транзакцию?").

Из особенностей следуют правила:

- Поскольку исключение не отменяет транзакцию сразу, но запрещает успешное завершение транзакции, то все вызовы НачатьТранзакцию с одной стороны и ЗафиксироватьТранзакцию или ОтменитьТранзакцию с другой стороны должны быть парными.
- Начало транзакции и ее фиксация (отмена) должны происходить в контексте одного метода
- Обработка исключений должна придерживаться следующих правил:
  - метод НачатьТранзакцию должен быть за пределами блока Попытка-Исключение непосредственно перед оператором Попытка
  - все действия, выполняемые после вызова метода НачатьТранзакцию, должны находиться в одном блоке Попытка, в том числе чтение, блокировка и обработка данных
  - метод ЗафиксироватьТранзакцию должен идти последним в блоке Попытка перед оператором Исключение, чтобы гарантировать, что после ЗафиксироватьТранзакцию не возникнет исключение
  - необходимо предусмотреть обработку исключений – в блоке Исключение нужно сначала вызвать метод ОтменитьТранзакцию, а затем выполнять другие действия, если они требуются
  - рекомендуется в блоке Исключение делать запись в журнал регистрации
  - при использовании вложенных транзакций в конце блока Исключение рекомендуется добавить оператор ВызватьИсключение. В противном случае исключение не будет передано выше по стеку вызовов, там не сработает обработка исключения, внешняя транзакция не будет явным образом отменена и платформа вызовет исключение «В данной транзакции происходила ошибка»

```
НачатьТранзакцию();
Попытка
    БлокировкаДанных = Новый БлокировкаДанных;
    ЭлементБлокировкиДанных =
БлокировкаДанных.Добавить("Документ.ПриходнаяНакладная");
    ЭлементБлокировкиДанных.УстановитьЗначение("Ссылка", СсылкаДляОбработки);
    ЭлементБлокировкиДанных.Режим = РежимБлокировкиДанных.Исключительный;
    БлокировкаДанных.Заблокировать();
```

```
    ... // чтение или запись данных
    ДокументОбъект.Записать();
    ЗафиксироватьТранзакцию();
Исключение
    ОтменитьТранзакцию();
    ЗаписьЖурналаРегистрации(НСтр("ru = 'Выполнение операции'"),
        УровеньЖурналаРегистрации.Ошибка,,,
        ОбработкаОшибок.ПодробноеПредставлениеОшибки(ИнформацияОбОшибке()));
    ВызватьИсключение; // есть внешняя транзакция
КонецПопытки;
```

- Использование вложенных транзакций приводит к усложнению кода. Принимая решение об использовании этой возможности, нужно очень взвешенно оценить решаемую задачу: возможно, это усложнение просто не оправдано.
  - Не стоит усложнять код, явно используя метод НачатьТранзакцию, когда кроме записи объекта другие действия с базой данных не делаются – платформа при записи сама откроет транзакцию.
  - Не нужно явно открывать транзакцию тогда, когда не требуется выполнять ответственное чтение данных. Например, обычно ответственное чтение не требуется при записи нового объекта (нового набора записей регистра).
  - При использовании методов ПолучитьОбъект (или Прочитать для наборов записей) необходимо анализировать должно ли чтение быть ответственным и в зависимости от этого принимать решение о явном использовании метода НачатьТранзакцию.
  - Если метод рассчитан на вызов только в рамках уже открытой транзакции (например, метод предназначен для вызова только из событий ПередЗаписью, ОбработкаПроведения и т.п.) в общем случае явным образом открывать в нем транзакцию не имеет никакого практического смысла.
  - При необходимости повысить качество сообщений об ошибках – на каждом уровне разработчик может предусмотреть свою обработку исключений, для чего, возможно, потребуется открыть вложенную транзакцию.
    Пример. Вызывается метод ДобавитьЭлектроннуюПодпись. Внутри, если что-то пошло не так, нужно обработать исключение и добавить текст вида: «Не удалось добавить электронную подпись к объекту %ПредставлениеОбъекта% по причине:%ОписаниеОшибки%». В противном случае исключение будет обработано выше по стеку вызовов, например, при записи файла и будет выдано сообщение вида: «Не удалось записать файл %ИмяФайла% по причине: %ОписаниеОшибки%», где в «%ОписаниеОшибки%», будет просто указание на строчку кода и пользователю будет непонятно, зачем вообще программа записывала файл, если он просто его подписывал.
  - При обработке исключения, если транзакция все еще активна, например, исключение возникло во вложенной транзакции, нельзя обращаться к базе данных, так как это приведет к исключению «В этой транзакции уже происходили ошибки». При этом нужно учитывать, что обращение к базе

данных может быть неявным, например, для получения представления ссылки.

- В общем случае в рамках одной транзакции нужно выполнять только те действия, которые неделимы, исходя из бизнес-логики.
  Пример. При проведении документа записывается документ и его движения в регистрах. Если не прошла запись хотя бы в один регистр вся операция проведения должна быть отменена.
- Следует избегать транзакций, которые выполняются длительное время.
  Например, неправильно: для загрузки адресного классификатора записывать все данные, относящиеся к одной версии классификатора в одной транзакции, для того, чтобы в случае ошибки откатить целиком загружаемую версию классификатора. Т.к. данных по одной версии классификатора много (объем около 1 Гб), то для выполнения такой транзакции, во-первых, может не хватить оперативной памяти (особенно при использовании файловой информационной базы на 32-разрядной ОС), а, во-вторых, такая операция будет выполняться достаточно долго и ее нельзя будет оптимизировать за счет выполнения в несколько потоков.
  Правильно: разбить загрузку новой версии классификатора на небольшие порции так, чтобы запись порции в одной транзакции не превышала 20 секунд в условиях высоконагруженной информационной системы и реализовать функциональность по откату к предыдущей версии в случае ошибки. Максимальная продолжительность указана исходя из того, что время ожидания установки транзакционной блокировки данных в информационной базе по умолчанию равно 20 сек.
- Чем дольше выполняется транзакция, тем большее время будут заняты ресурсы сервера 1С:Предприятия и СУБД. Как правило длинные транзакции занимают следующие ресурсы:
  - в ходе выполнения транзакции все изменения в базе данных записываются в журнал транзакций, что необходимо для возможности откатить транзакцию;
  - блокировки, установленные в транзакции, остаются до конца транзакции
  - на сервере 1С:Предприятия блокировки занимают оперативную память
  - другие ресурсы, необходимые самой бизнес-логике, которая выполняется в транзакции.
- Если две транзакции пересекаются по блокируемым ресурсам, то транзакция, которая начала выполняться позже, будет ожидать возможность установления блокировки ограниченное время (по умолчанию – 20 секунд), после чего будет завершена с исключением «Превышено время ожидания установки блокировки». Поэтому длинные транзакции могут сильно снижать удобство параллельной работы пользователей.
  Возникновение таких исключений – это повод провести анализ действий, которые выполняются в конфликтующих транзакциях
- В рамках транзакции нужно стремиться выполнять минимум действий – только те, которые нельзя в соответствии с бизнес-логикой выполнять вне транзакции.
- Обязательное использование транзакции в случае, если для ускорения операции записи в регистр используется отключение итогов, такую операцию вместе с отключением и включением итогов необходимо выполнять в транзакции, иначе в других сеансах может возникнуть ошибка при получении среза последних.

Проверка понимания.

Определение и граница использования транзакции. Транзакции - нечто неделимое, возвращающее все в предыдущее состояние. Это осталось в голове после заумных текстов без тестов лапками. Первым, что пришло в голову - восстановить значение реквизита :) Обработка с полем ввода ТестовоеЧисло, 0 по умолчанию.

```
&НаСервере
Процедура БезОбработкиОшибокНаСервере()
НачатьТранзакцию();
ТестовоеЧисло = 5;
ВызватьИсключение "Число не то!";
ЗафиксироватьТранзакцию();
КонецПроцедуры

&НаКлиенте
Процедура БезОбработкиОшибок(Команда)
БезОбработкиОшибокНаСервере();
КонецПроцедуры
```

Не удалось! Ошибка сгенерировалась, но ТестовоеЧисло стало 5. Ведь транзакции актуальны при изменении данных в базе, т е SQL запрос обрамляется чем-то типа "BEGIN ... COMMIT". Мой косяк, поехали дальше.

Ошибка при создании элемента справочника. Справочник Города.

```
&НаСервереБезКонтекста
Процедура НовыйЭлементСправочникаНаСервере()
НачатьТранзакцию();
НовыйГород = Справочники.Города.СоздатьЭлемент();
НовыйГород.Наименование = "Тестовый город";
НовыйГород.Записать();
    НовыйГород2 = Справочники.Города.СоздатьЭлемент();
НовыйГород2.Наименование = "Тестовый город 2";
НовыйГород2.Записать();
ВызватьИсключение "Город не правильный!";
ЗафиксироватьТранзакцию();
КонецПроцедуры
```

Да, тестовые города не создались. Получилось!

Вложенные транзакции (или то чего нет).

```
&НаСервереБезКонтекста
Процедура ВложеннаяТранзакция()
	НачатьТранзакцию();
	Попытка
		НовыйГород = Справочники.Города.СоздатьЭлемент();
		НовыйГород.Наименование = "Тестовый город вложенный";
		НовыйГород.Записать();
		ВызватьИсключение "Город не правильный!";
		ЗафиксироватьТранзакцию();
	Исключение
		ОтменитьТранзакцию();
	КонецПопытки;
КонецПроцедуры

&НаСервереБезКонтекста
Процедура ВложенныеТарнзакцииНаСервере()
	НачатьТранзакцию();
	НовыйГород = Справочники.Города.СоздатьЭлемент();
	НовыйГород.Наименование = "Тестовый город внешний";
	НовыйГород.Записать();
	ВложеннаяТранзакция();
	ЗафиксироватьТранзакцию();
КонецПроцедуры
```

Во вложенной транзакции мы подавляем исключение - и вся транзакция тихо не выполняется. Ни один город не создан. Я не получил вообще никакого отклика от платформы, что и логично. Теперь попробуем добавить во внешний код обработку ошибок.

```
&НаСервереБезКонтекста
Процедура ВложеннаяТранзакция()
	НачатьТранзакцию();
	Попытка
		НовыйГород = Справочники.Города.СоздатьЭлемент();
		НовыйГород.Наименование = "Тестовый город вложенный";
		НовыйГород.Записать();
		ВызватьИсключение "Город не правильный!";
		ЗафиксироватьТранзакцию();
	Исключение
		ОтменитьТранзакцию();
	КонецПопытки;
```

```
КонецПроцедуры

&НаСервереБезКонтекста
Процедура ВложенныеТарнзакцииНаСервере()
НачатьТранзакцию();
Попытка
НовыйГород = Справочники.Города.СоздатьЭлемент();
НовыйГород.Наименование = "Тестовый город внешний";
НовыйГород.Записать();
ВложеннаяТранзакция();
ЗафиксироватьТранзакцию();
Исключение
ОтменитьТранзакцию();
Сообщить("Исключение было поймано");
КонецПопытки;
КонецПроцедуры
```

Вполне логично, что сообщение не было сформировано.

Проверка функции ТранзакцияАктивна.

```
&НаСервереБезКонтекста
Процедура НовыйЭлементСправочникаНаСервере()
Сообщить(ТранзакцияАктивна());
НачатьТранзакцию();
НовыйГород = Справочники.Города.СоздатьЭлемент();
НовыйГород.Наименование = "Тестовый город 3";
НовыйГород.Записать();
Сообщить(ТранзакцияАктивна());
ЗафиксироватьТранзакцию();
Сообщить(ТранзакцияАктивна());
КонецПроцедуры
```

Ожидаемый вывод: Нет, Да, Нет. Теперь проверим факт из статьи о блокировках о возникновении блокировки и как следствие транзакции при чтении (это так я понял текст)

```
&НаСервереБезКонтекста
Процедура ТранзакцияПриЧтенииНаСервере()
МойГород = Справочники.Города.НайтиПоНаименованию("Иркутск");
Сообщить(ТранзакцияАктивна());
НовыйГород = Справочники.Города.СоздатьЭлемент();
```

```
	НовыйГород.Наименование = "Тестовый город 3";
	НовыйГород.Записать();
	Сообщить(ТранзакцияАктивна());
КонецПроцедуры

&НаКлиенте
Процедура ТранзакцияПриЧтении(Команда)
	ТранзакцияПриЧтенииНаСервере();
КонецПроцедуры
```

И получил Нет, Нет. Видимо что-то не так понял. Аналогичный результат при поэлементном чтении всего справочника. После внимательного рассмотрения, о транзакции в пределах процедуры было сказано для обработки события ПриЗаписиНаСервере (в случае сохранения данных через форму). Чтож, проверим это. Если транзакция есть, то данный код не запишет данные.

```
&НаСервере
Процедура ПриЗаписиНаСервере(Отказ, ТекущийОбъект, ПараметрыЗаписи)
	Отказ = ТранзакцияАктивна();
КонецПроцедуры
```

И да, создать объект не удалось, значит транзакция была активна!

Количество выполненных действий при исключении во вложенной транзакции.
Смоделируем ситуацию: при входе во вложенную транзакцию происходит подавленная ошибка. Причем только одна ошибка в середине процесса. Сколько раз выполнится внешняя транзакция? Ведь в одном из предыдущих примеров исключение из вложенной транзакции поймано не было.

```
&НаСервереБезКонтекста
Процедура ВложеннаяТранзакция()
	НачатьТранзакцию();
	Попытка
		НовыйЭлемент = Справочники.ДляТранзакций.СоздатьЭлемент();
		НовыйЭлемент.Наименование = "Тестовый элемент внутренний";
		НовыйЭлемент.Записать();
		ВызватьИсключение "Ошибка!";
		ЗафиксироватьТранзакцию();
	Исключение
		ОтменитьТранзакцию();
		Сообщить("Ошибка во вложенной транзакции");
	КонецПопытки;
```

```
КонецПроцедуры

&НаСервереБезКонтекста
Процедура ТестТранзакцииНаСервере()
	НачатьТранзакцию();
	Попытка
		Для сч = 1 По 5 Цикл
			НовыйЭлемент = Справочники.ДляТранзакций.СоздатьЭлемент();
			НовыйЭлемент.Наименование = "Тестовый элемент внешний " + Строка(сч);
			НовыйЭлемент.Записать();
			Сообщить("Попытка записи " + Строка(сч) + " внешнего элемента.");
			//Если сч = 3 Тогда
			//	ВложеннаяТранзакция(); //ошибка в середине
			//КонецЕсли;
		КонецЦикла;
		ВложеннаяТранзакция();//ошибка в конце
		ЗафиксироватьТранзакцию();
	Исключение
		ОтменитьТранзакцию();
		Сообщить("Исключение было поймано");
	КонецПопытки;
КонецПроцедуры
```

Сначала все логично. В указанном варианте (он повторяет предыдущий эксперимент) выведено 5 сообщений и "Ошибка во вложенной транзакции". Сообщение "Исключение было поймано" не отображается.

А дальше ждал сюрприз. Если раскомментировать условие (ошибка на третьем шаге) то получается какая-то бабуйня: выведено 3 сообщения, сообщение "Ошибка во вложенной транзакции" и сообщение "Исключение было поймано"! То есть исключение транслируется наверх, если происходит между выполняемыми действиями. Однако если после выполняемых действий, но до фиксации транзакции - ошибка не транслируется наверх, а транзакция тихо откатывается.

Revision #5
Created 12 February 2025 13:58:14 by Admin
Updated 13 February 2025 03:47:20 by Admin