Manipulação do Commit e Níveis de isolamento

GeneXus™

Os objetos GeneXus do tipo Transação e Procedimento oferecem a propriedade Commit on exit que pode assumir o valor Yes ou No. Desta forma, os programas gerados executarão um Commit automático ou não.

Propriedade Commit on Exit

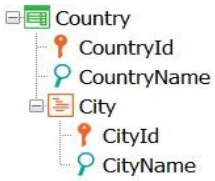
Commit on Exit = Yes

Procedimentos

```
For each Country
     Where CountryId = 2
     CountryName = "New country name"
endfor
```

É incluído um Commit automático no final do source nos programas gerados.

Transações



É incluído um Commit automático nos programas gerados, após realizar uma modificação na base de dados.

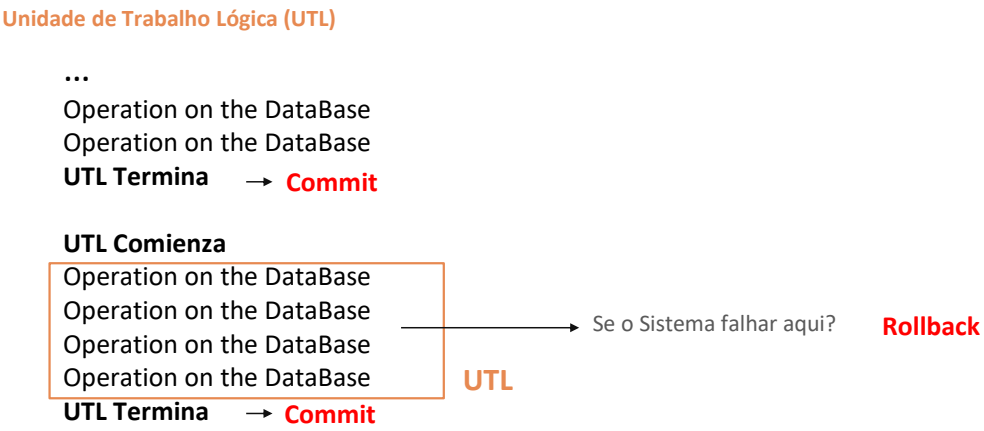
Por padrão, GeneXus inclui um comando Commit nos programas gerados associados aos objetos do tipo Transação e Procedimento.

- Nos Procedimentos, é incluído um Commit automático no final do Source nos programas gerados.
- Nas Transações é incluído um Commit automático nos programas gerados, após efetuar uma modificação na base de dados. Ou seja, após inserir, modificar ou excluir dados e imediatamente **antes** de executar o código associado às regras condicionadas no momento de disparo AfterComplete e o código associado ao evento AfterTrn.

Por exemplo, se tiver a transação Country, com City como segundo nível, e forem inseridos 5 países através do seu form, o Commit será executado 5 vezes, depois de gravada a informação de cada país e suas cidades, mas antes da execução das regras condicionadas no momento AfterComplete.

Propriedade Commit on Exit

Que motivos podem existir para não executar um Commit em uma Transação ou em um Procedimento?

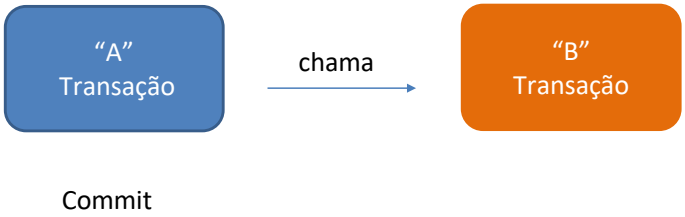


Que motivos podem existir para não executar um Commit em uma transação ou em um procedimento?

Para personalizar uma Unidade de Trabalho Lógica (LUW – Logical Unit of Work)

Isto significa que seja necessário expandir uma UTL, de forma que, por exemplo, vários procedimentos, ou uma transação com um ou vários procedimentos, componham uma unidade de trabalho lógica, e seja necessário que um Commit englobe a todos eles

Restrições



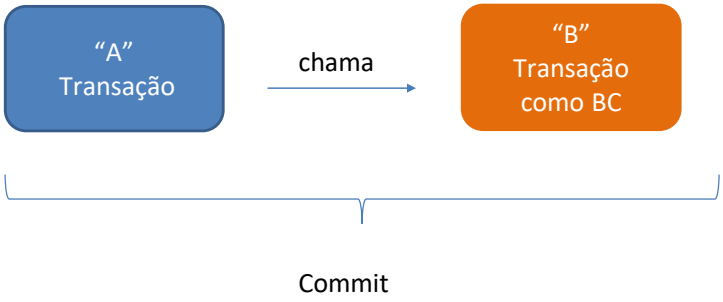
Commit

Vejamos agora algumas restrições importantes em relação a este tema

Em ambientes web, cada transação pode commitar apenas seu próprio conjunto de operações realizadas na base de dados, e dos procedimentos invocados por ela, mas não as operações realizadas por outra transação.

Ou seja, se uma transação chama outra transação, o Commit executado por uma transação não se aplica aos registros inseridos, modificados ou excluídos pela outra. Portanto, duas transações diferentes não podem ser incluídas em uma mesma UTL

Restrições



A propriedade Commit on Exit será ignorada em transações utilizadas como Business Component.

Caso necessite executar operações através de duas transações diferentes, e deseje compor entre elas apenas uma UTL, a solução é executá-las utilizando o conceito de Business Component, incluindo, então, o comando Commit após executar as operações associadas a ambas as transações.

Vale destacar que a propriedade Commit on exit será ignorada em transações utilizadas como Business Component.

Isto significa que, embora a transação tenha a propriedade Commit on exit com o valor Yes, se a mesma for utilizada como Business Component, o commit não será executado de forma automática e será necessário declarar o comando commit de forma explícita.

O motivo deste comportamento é permitir especificar unidades de trabalho lógica entre múltiplas transações, incluindo o comando commit onde seja necessário.

Propriedade Commit on Exit - Exemplos

1

Commit on Exit = Yes


```
For each Country
     Where CountryId = 2
     CountryName = "New country name"
endfor
```

A propriedade Commit on Exit será considerada e GeneXus adicionará o Commit de forma automática.

Vamos então propor quatro exemplos muito específicos e analisaremos o comportamento:

Vejamos então o primeiro exemplo:

Suponhamos a transação Country e o source do procedimento mostrado. O procedimento tem a propriedade Commit on Exit com o valor Yes.

Como o For each realiza uma atualização na base de dados, GeneXus adicionará o commit automaticamente e o país com valor CountryId = 2 será atualizado com seu novo nome.

Propriedade Commit on Exit - Exemplos

2

Business Component = Yes

Commit on Exit = Yes

```
&Country.CountryName = "Brazil"
&Country.Insert()
```

A propriedade Commit on Exit será ignorada e o país não será commitado.

Vejamos o seguinte:

Suponhamos a mesma Country declarada como Business Component, e o source do procedimento mostrado:

O atributo CountryId é autonumerado e o procedimento tem a propriedade Commit on Exit com o valor Yes.

Que comportamento terá?

Como o procedimento somente realiza operações com o Business Component, mesmo que o procedimento tenha a propriedade Commit on Exit com o valor Yes, será ignorada e o país não será commitado. Para conseguir isso, é necessário adicionar o comando Commit de forma explícita.

Commit on Exit property - Examples

3

Business Component = Yes

Commit on Exit = Yes

```
&Country.CountryName = "Brazil"
&Country.Insert()

For each Country
     Where CountryId = 2
     CountryName = "New country name"
endfor
```

A propriedade Commit on Exit será considerada e tanto as operações do BC como as do For each serão commitadas.

Passemos para o seguinte exemplo:

Suponhamos outra vez a mesma transação Country declarada como Business Component, e o source do procedimento mostrado:

O atributo CountryId é autonumerado e o procedimento tem a propriedade Commit on Exit com o valor Yes

Neste exemplo, ao ter operações com um Business Component no source de um procedimento, e ter também um For each que realiza atualizações na base de dados, então sim, é considerado o valor Yes da propriedade Commit on Exit e são commitadas tanto as operações com o Business Component como as operações do For each.

Portanto, será inserido o país chamado Brasil e também o país com valor CountryId = 2 mudará seu nome.

Commit on Exit property - Examples

4

Business Component = Yes

Commit on Exit = Yes

```
For each Country
     Where CountryId < 3
     &Country.Load(CountryId)
     &Country.CountryName = "New country name"
     &Country.Update()
endfor
```

A propriedade Commit on Exit será ignorada.

Vejamos o último exemplo:

Novamente, consideramos a mesma transação Country declarada como Business Component, e o source do procedimento mostrado:

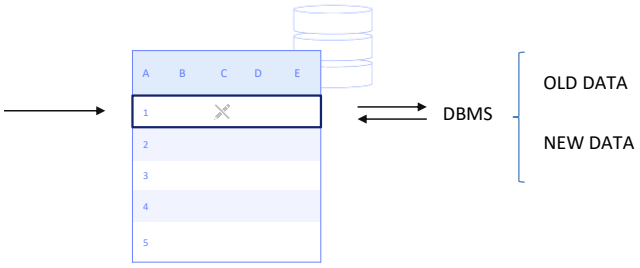
O atributo CountryId é autonumerado e o procedimento tem a propriedade Commit on Exit com o valor Yes.

Neste caso, o valor Yes da propriedade Commit on Exit também será ignorado, pois a atualização que se tenta fazer é através do Business component e não diretamente pelo For each. Se este Business Component não estivesse presente, o For each sozinho não realiza nenhuma ação.
Portanto, será necessário adicionar o comando Commit de forma explícita.

Então, depois de ver estes exemplos, como regra geral, podemos dizer que um objeto com a propriedade Commit on Exit em Yes fará Commit ao finalizar, somente se o referido objeto realiza alguma atualização na base de dados que não seja somente através de um Business Component.

Níveis de isolamento

Read Only



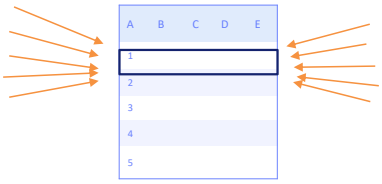
Bem. Vamos ver algo mais:

Embora não vamos nos concentrar agora em estudar o controle de concorrência, vale mencionar que quando vários usuários realizam operações na base de dados, se a informação a ser lida estiver bloqueada por outro programa de escrita, os valores que serão exibidos dependerão do DBMS. É o DBMS então que decidirá se mostra o valor antigo ou o novo dos dados que estão sendo consultados.

Especificar o **nível de isolamento** afeta a leitura e o controle de concorrência

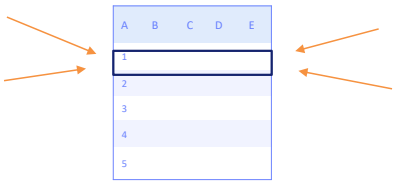
Níveis de isolamento

Low isolation level



Increases simultaneous access effects

High isolation level



Reduces simultaneous access effects

Nos sistemas de bases de dados, o isolamento determina a forma que a integridade das transações na base de dados é visível para outros usuários e sistemas.

- Um nível de isolamento mais baixo aumenta a capacidade de muitos usuários acessarem os mesmos dados ao mesmo tempo, mas aumenta também a quantidade de efeitos de simultaneidade, como por exemplo as leituras sujas ou atualizações perdidas, que os usuários podem encontrar. Também aumenta a possibilidade de bloqueio, o que requer técnicas muito cuidadosas de análise e programação para evitá-lo.

- Pelo contrário, um nível de isolamento mais alto reduz os tipos de efeitos de simultaneidade que podem encontrar os usuários, mas requer mais recursos do sistema e aumenta as possibilidades de que uma transação de base de dados bloqueie outra. O DBMS geralmente adquire bloqueios nos dados que podem resultar em uma perda de concorrência ou implementa um controle de concorrência de múltiplas versões. Isto requer adicionar lógica para que a aplicação funcione corretamente.

Níveis de isolamento

Os níveis de isolamento são os seguintes:

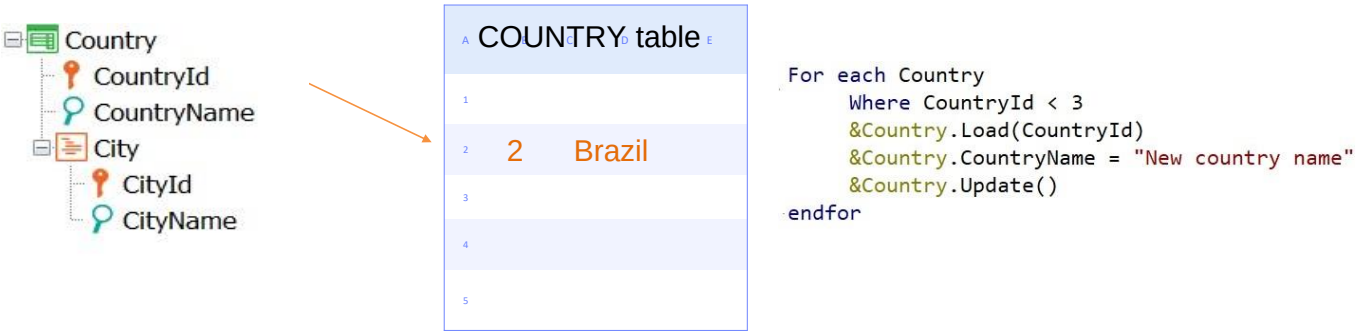
- **O nível Serializável**: É o nível de isolamento mais alto. Requer que os bloqueios de leitura e escrita (adquiridos em dados selecionados) se liberem ao final da transação. Quando é utilizado o controle de concorrência não baseado em bloqueios, se o sistema detectar uma colisão de gravação entre várias transações simultâneas, apenas uma delas poderá ser confirmada.

- **Leituras repetidas**: Neste nível de isolamento, uma implementação DBMS de controle de concorrência baseada em bloqueios, mantém os bloqueios de leitura e gravação até o final da transação. No entanto, os bloqueios de intervalo não são gerenciados, portanto, podem ocorrer leituras fantasmas.

- **Ler confirmados:** É um nível de isolamento que garante que qualquer leitura de dados seja confirmada no momento em que for lida. Simplesmente impede que o leitor veja qualquer leitura intermediária, não confirmada e 'suja'.

- **O nível Ler não confirmados:** É o nível de isolamento mais baixo. São permitidas leituras sujas, portanto, uma transação de base de dados pode ver alterações realizadas por outras transações que ainda não foram confirmadas.

Níveis de isolamento - Exemplo



If Isolation level is Read Committed – CountryName = “Brazil”

If Isolation level is Read Uncommitted – CountryName = “New country name”

DIRTY READ

Vejamos um exemplo para ilustrar estes conceitos:

Vamos considerar a tabela COUNTRY.
Um usuário acessa o registro 2 e obtém Brazil como valor do atributo CountryName
Em seguida, executa um código como o que estamos vendo.

O que devemos considerar? Que por ser um Business Component e não ter sido escrito explicitamente o commit na base de dados, fisicamente é mantido o nome "Brazil" até que efetivamente seja executado um Commit.

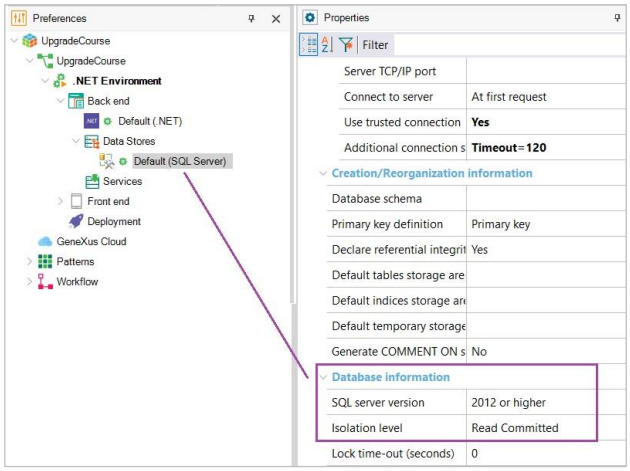
Agora, antes que seja executado um commit entra outro usuário para ler o registro 2 e é então que dependendo do nível de isolamento obterá diferentes valores

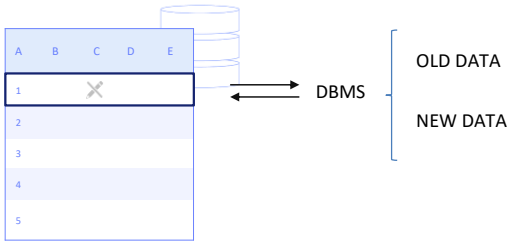
- Se o nível de isolamento for Read Committed, então obtém o valor “Brazil” para o atributo CountryName
- Se o nível de isolamento for Read Uncommitted, então obtém “New Country Name” como valor do atributo CountryName

Então:
Read Uncommited (Ler não confirmados) é um nível baixo de isolamento, enquanto Read Committed (Ler confirmados) é um nível alto.
Associado ao nível Read Uncommitted está o conceito “Dirty read”, “leitura suja”, que justamente se refere à possibilidade de ler dados que ainda não foram confirmados.

Quanto aos níveis Serializable e Repeat Reading, são ainda mais restritivos, pois adicionam um lock, ou seja, que o novo usuário nem poderá acessar o registro 2 porque estará bloqueado

GeneXus – Isolation Level property





Em GeneXus, a propriedade Isolation Level, localizada no nível do Data Store, permite especificar o nível de isolamento das mudanças realizadas pelos programas.

Os valores possíveis são os seguintes:

- **Read committed:** Isto significa que os programas não veem as alterações realizadas por outros usuários até que seja executado o commit. Este é o valor padrão que assume esta propriedade.
- **Read Uncommitted:** Neste caso, os programas veem as alterações realizadas por outros usuários mesmo quando ainda não foi executado o Commit.

Por tudo isto que vimos, especificar o nível de isolamento afeta a leitura dos dados e o controle de concorrência.

training.genexus.com
wiki.genexus.com
training.genexus.com/certifications