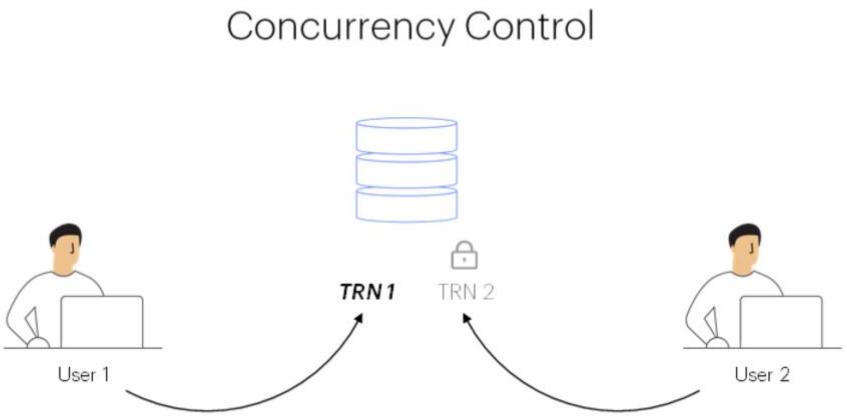
Concurrency Control

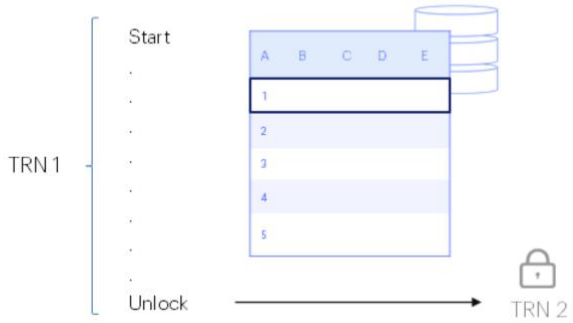
GeneXus™



When we talk about Concurrency Control we make reference to a set of controls to avoid possible inconsistencies in the data when working in environments with multiple users.

To control concurrency it is necessary to block the information.

Concurrency Control



In addition, the concurrent execution of programs should not cause poor performance.

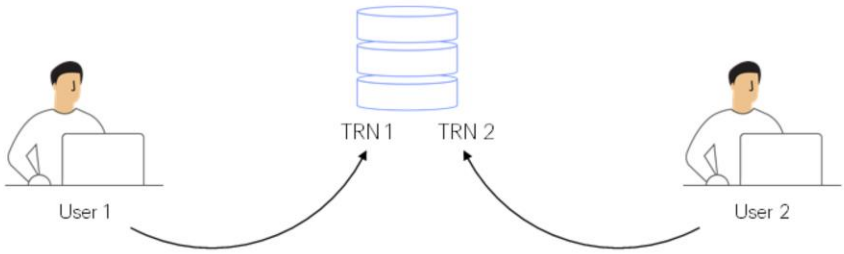
Concurrency Control
Optimistic

Transactions

Business Components

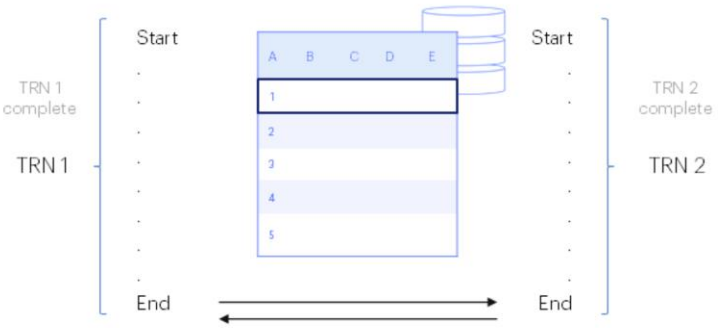
Let's start by looking at concurrency control in Transactions and Business Components. Let's talk then about “optimistic concurrency control.”

Optimistic Concurrency Control



This mechanism assumes that multiple database transactions can be completed without affecting each other, and that therefore these transactions can be performed without blocking the data resources involved.

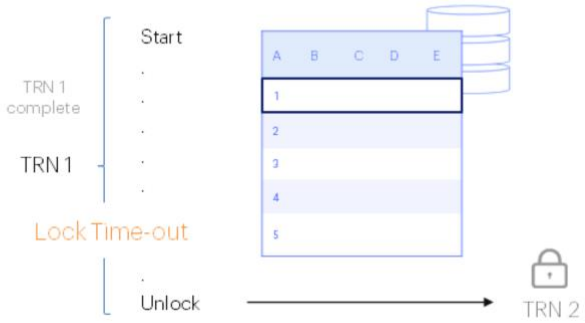
Optimistic Concurrency Control



Before performing an update, each database transaction checks that no other database transaction has modified its data.

This is related to the globalized nature of the web where several users can access the same page at the same time. This makes blocking unfeasible for web user interfaces.

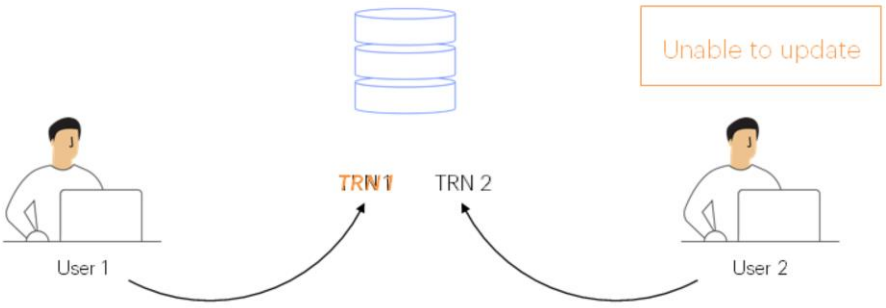
Optimistic Concurrency Control



It is common for a user to start editing a record and then leave without following the “cancel” or “log out” link.

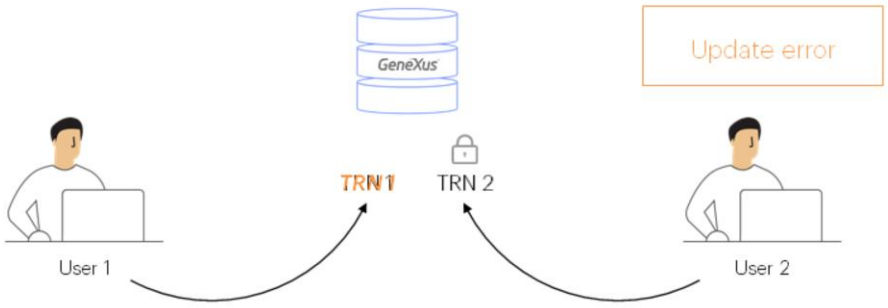
If locking is used, other users trying to edit the same record must wait until the first user's lock time expires.

Optimistic Concurrency Control



Instead of locking each record every time it is used, the system simply looks for signs that two users actually tried to update the same record at the same time. If such evidence is found, then a user's updates are discarded and the user is informed of this.

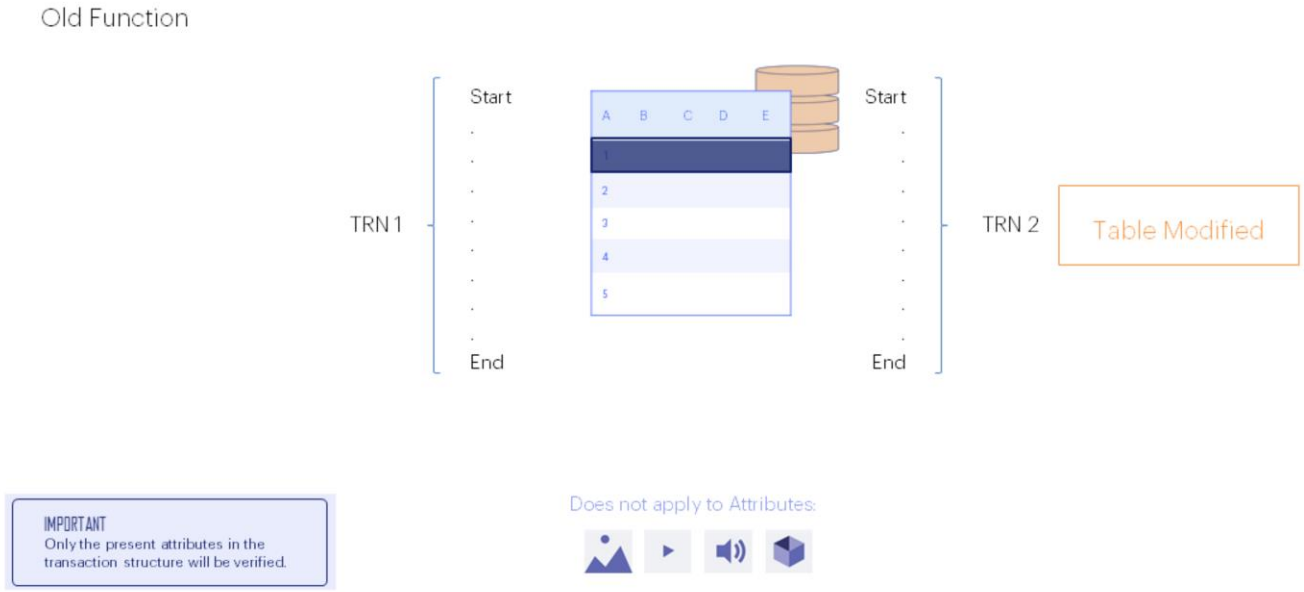
Optimistic Concurrency Control



In GeneXus, when two or more users want to update the same record, the first one will be able to update it and the others will get an error (because they were about to save new data based on outdated information that was updated by another user).

Then, in these cases, the confirmation operation is reversed.

Optimistic Concurrency Control



The optimistic concurrency control is based on the "Old Function."

When a record is confirmed, the "old" values of each attribute are compared to the current values in the database. If a value doesn't match, an error similar to the following is displayed: "The table has been modified," which means that another user has modified the record since the values were obtained.

It should be mentioned that:

- Only the attributes present in the transaction structure are taken into account. In other words, if there are parallel transactions with more attributes, they are not verified.
- This mechanism does not apply to attributes of types such as image, video, audio, Longvarchar, nor to those attributes inferred from the extended table that are updated by means of rules declared in the transaction.
- It only applies to attributes that are not part of the key, because the record is instantiated with them.

Concurrency Control in GeneXus Objects

when Reading Data

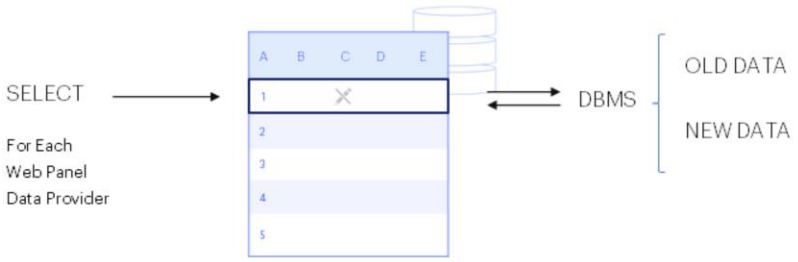
when Reading & Writing Data

Let's now see what concurrency control looks like in other GeneXus objects.

For example, how GeneXus handles locks when only reading data and when reading and writing data to control concurrency.

Concurrency Control in GeneXus Objects

Read Only



In GeneXus:
Isolation level Property

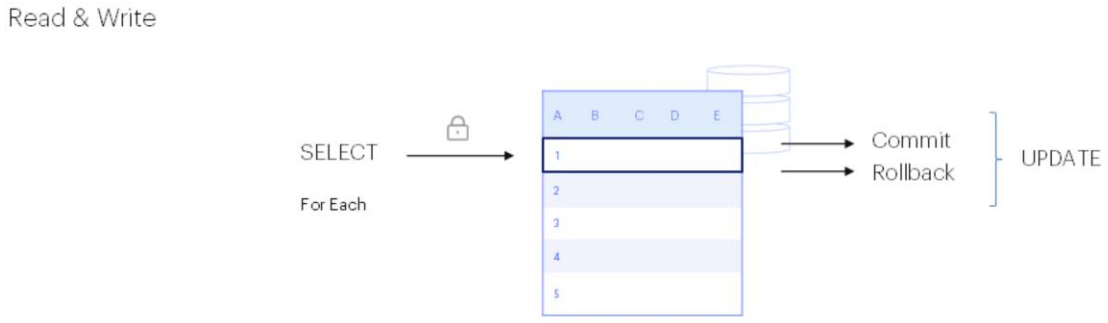
- Read Committed
- Read Uncommitted

What happens when data is only read?

If it is read-only (when using For Each commands, Web Panel type objects, Data Providers, etc.), the generated SELECTs are NOT locked.
They are always affected by exclusive locks. For example, in reorganizations, tables are opened in an exclusive way. In this case, no other process will be able to open the table, no matter if it is read-only.

On the other hand, if the information to be read is locked by another write program, the values to be displayed will depend on the DBMS. It is then up to the DBMS to decide whether to display the old or the new value.

Concurrency Control in GeneXus Objects



IMPORTANT
Certain For Each commands are optimized and won't lock down. They directly update.

And what happens when data is read and written?

A For Each command that includes an update to the database, performs a SELECT with a lock when entering the For Each and then, the update is performed.

As any lock, it releases the records when executing the Commit or Rollback.

It should be noted that certain For Each commands are optimized and do not lock, but update directly.
Only For Each commands that contain conditions that go to the server, and only assignments in their body, can be optimized.

GeneXus™

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