SQL Transaction Handling

The xUML Runtime works with the concept of sessions. Sessions are equivalent to units of work, that can be committed or rolled back depending on the status at the end of the session. Database access is also part of this concept and affected on session commit/rollback.
Refer to xUML Runtime Architecture and Transaction Concept for a detailed explanation of the xUML Runtime session and transaction concept.

In the context of the Designer, a session may be for example an **On Exit** execution of a service task or a **Get Data** execution of a user task.

- The Runtime implicitly commits a session when it is terminated without any exception, and thus, all involved transaction resources like e.g. an open SQL connection.
- If an exception occurred during session execution and if this exception is not being caught, the session gets implicitly rolled back. The xUML Runtime will then rollback all not committed transactions.

On this Page:

Related Documentation:

- xUML Runtime Architecture and Transaction Concept

SQL Commit

Sometimes a certain state of the database is considered good and you want to explicitly commit the changes. This is the case, e.g. when you are updating database records in a loop and you want to commit each record separately.
To explicitly commit changes to a database, specify the **commit** command in the **sql** attribute of the **exec ute** SQL adapter action.

SQL Rollback

Sometimes a certain state of the database is considered and you want to explicitly commit the changes. This is the case, e.g. when you are updating dependent tables and one update fails. Now you want to rollback all previous changes to other tables.
To explicitly rollback changes to a database, specify the **rollback** command in the **sql** attribute of the **exe cute** SQL adapter action.