

Basics of PHP - OCI function `oci_commit`

From CS 325, you should remember that the SQL command:

```
commit;
```

...is supposed to make the current database state PERMANENT. SQL provides this, along with `rollback;`, to support TRANSACTIONS, logical units of work, completely done or completely NOT done.

You should also remember that Oracle **sqlplus** does some auto-commits on your behalf:

- whenever the database structure changes, and
- when you log out "properly" from **sqlplus**.

So, if you haven't explicitly typed **commit;** before exiting your **sqlplus** session, the assumption is that you want that session's changes to be committed.

QUESTION, then: on the application tier, WHEN should changes be committed?

Java's JDBC (Java Database Connectivity) library assumes that each individual change is to be auto-committed unless you indicate otherwise. But this assumption means you cannot use **rollback** to undo multiple changes that may have been made by a transaction-in-progress if it runs into problems.

With OCI, it appears that this is determined by the second argument to `oci_execute`. When you use **OCI_DEFAULT** as `oci_execute`'s second argument,, this specifies to **NOT** auto-commit after each individual change, but to instead wait for a commit to be explicitly requested. This is a better choice for supporting transaction integrity.

OCI provides a function `oci_commit` to make this explicit commit request. It expects a connection object, and has the side-effect of requesting that a commit be done. For example:

```
oci_commit($conn);    // assuming $conn contains a connection object  
                      //          as returned by oci_connect
```

So: if you have a PHP document using OCI, and it requests any SQL actions that change the database (such as an **insert**, **update**, or **delete**), that PHP document should call `oci_commit` once it believes the current logical transaction is successfully completed and reasonable to commit.

Oh, and if you forget? Then that PHP document's changes are **NOT** committed to the database...!

(And, OCI also provides a function `oci_rollback`, that also expects a connection object, and has the side-effect of requesting that a rollback be done to the last previous committed state.)