

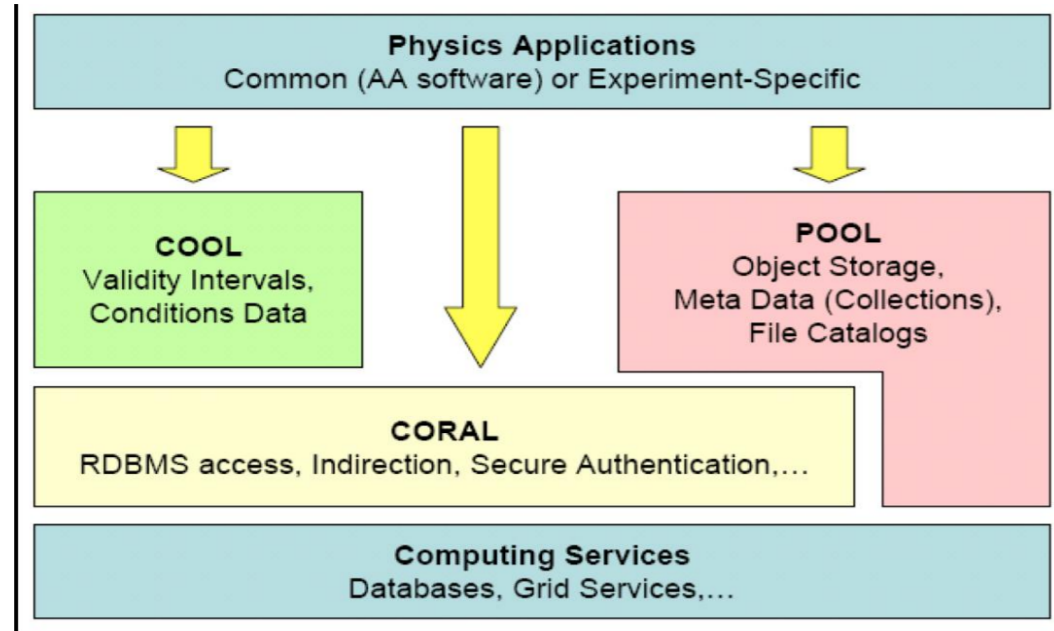
# Reacting to database and network instabilities in CORAL

✓ A distributed analysis model implies an in-homogeneity in the data storage infrastructure across the different institutes and across the long LHC lifetime.

✓ The computing infrastructure has to be easily maintainable and adaptable.

✓ POOL is a generic hybrid store for C++ objects, metadata catalogues and collections, using streaming and relational technologies.

✓ COOL provides specific software to handle the time variation and versioning of conditions data.



✓ CORAL is a generic abstraction layer with an SQL-free API for accessing relational databases.

CORAL provides a set of C++ libraries for several database back-ends:

- ✓ local access to SQLite files;
- ✓ direct client access to Oracle and MySQL servers;
- ✓ read-only access to Oracle through the FroNTier/Squid or CoralServer/CoralServerProxy server/cache system.

Why CORAL ?

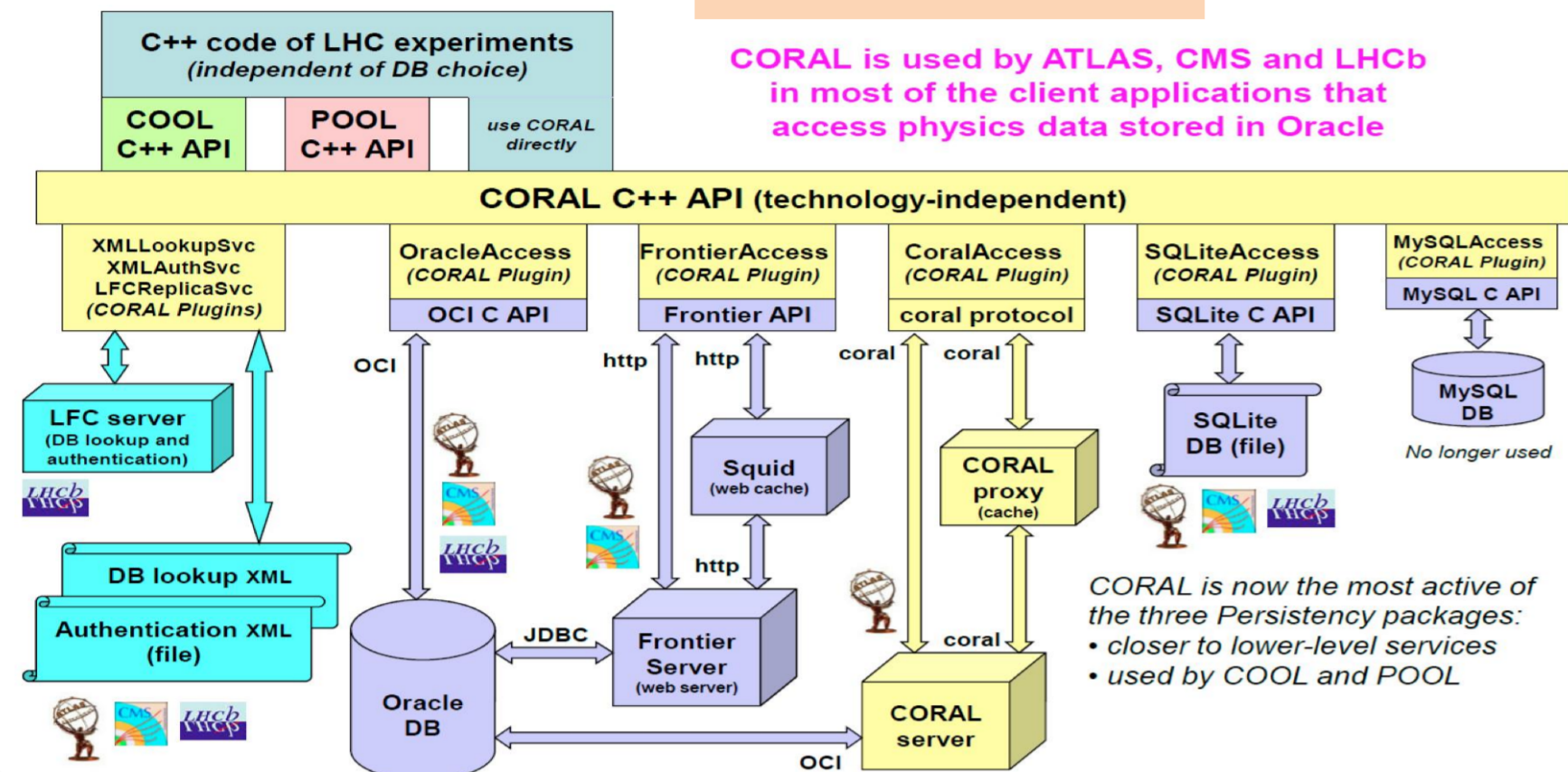


✓ Users write the same code for all back-ends

✓ A detailed knowledge of the many SQL flavors is not required

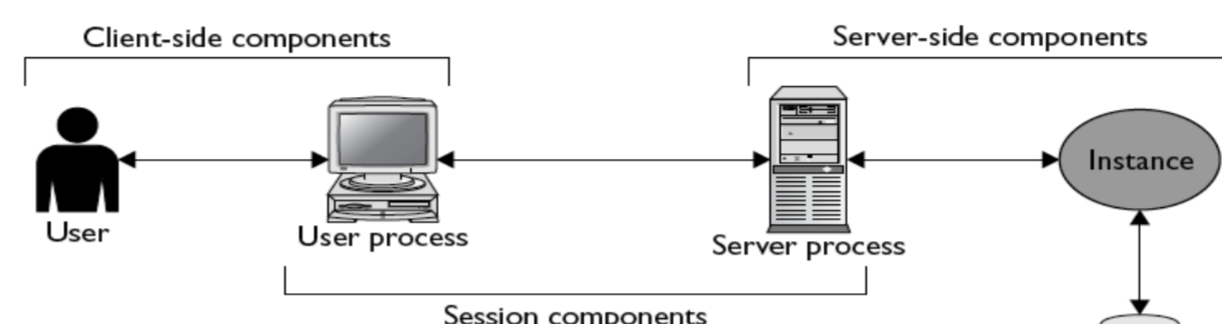
✓ The SQL commands specific to each backend are executed by the relevant CORAL libraries, which are loaded at run-time by a special plugin infrastructure.

## CORAL BACK-ENDS

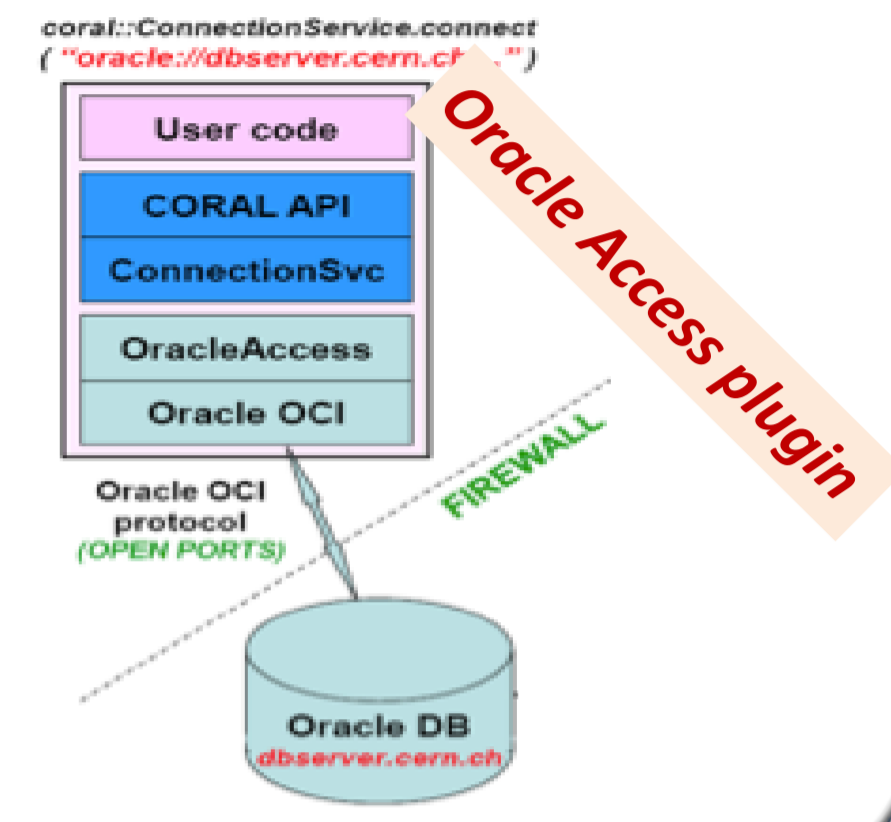


A **connection** is a "physical circuit", a pathway to a database.

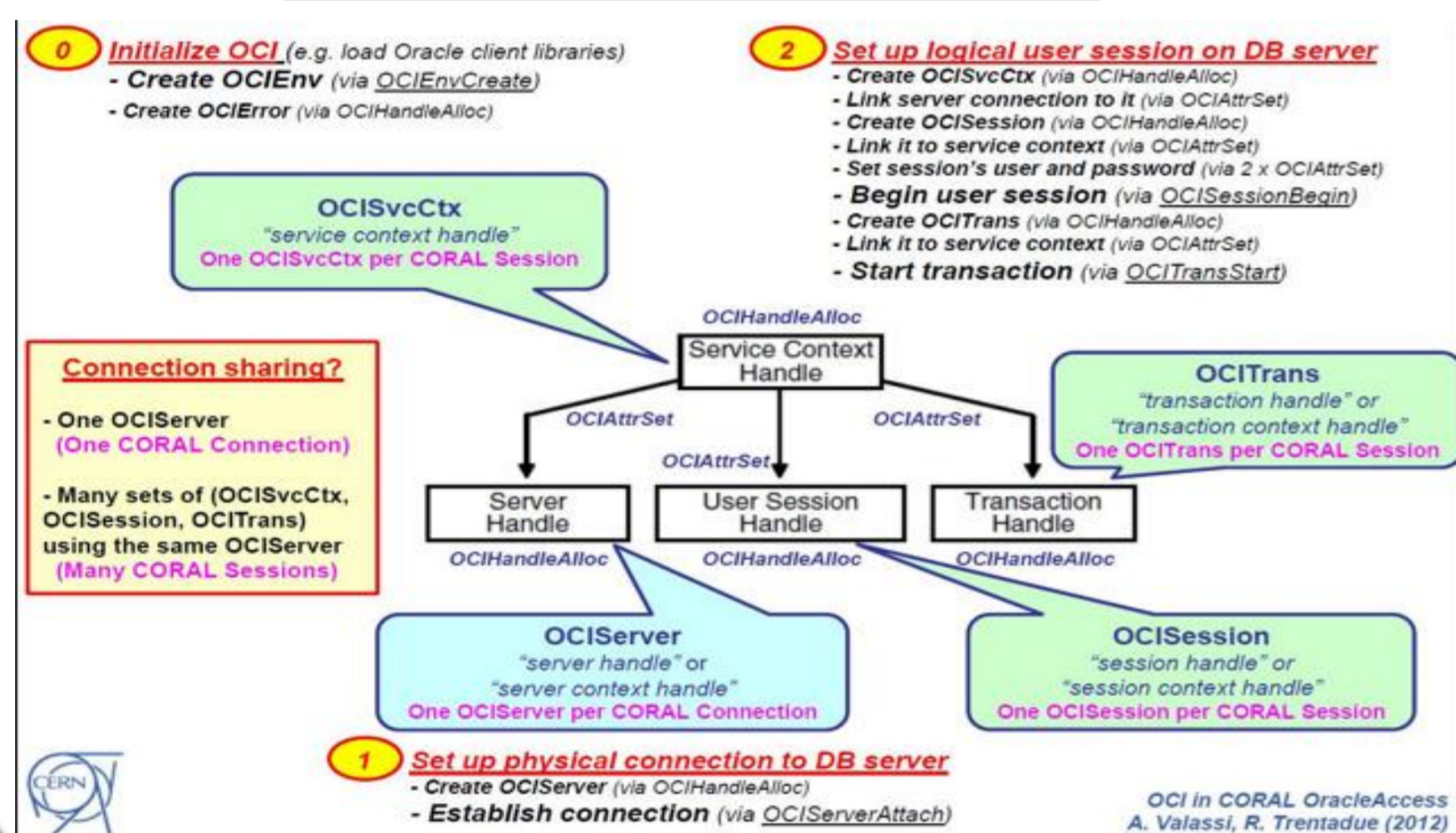
A **session** is a user process in communication with a server process. Users establish sessions against the instance, and the instance then manages the access to the database.



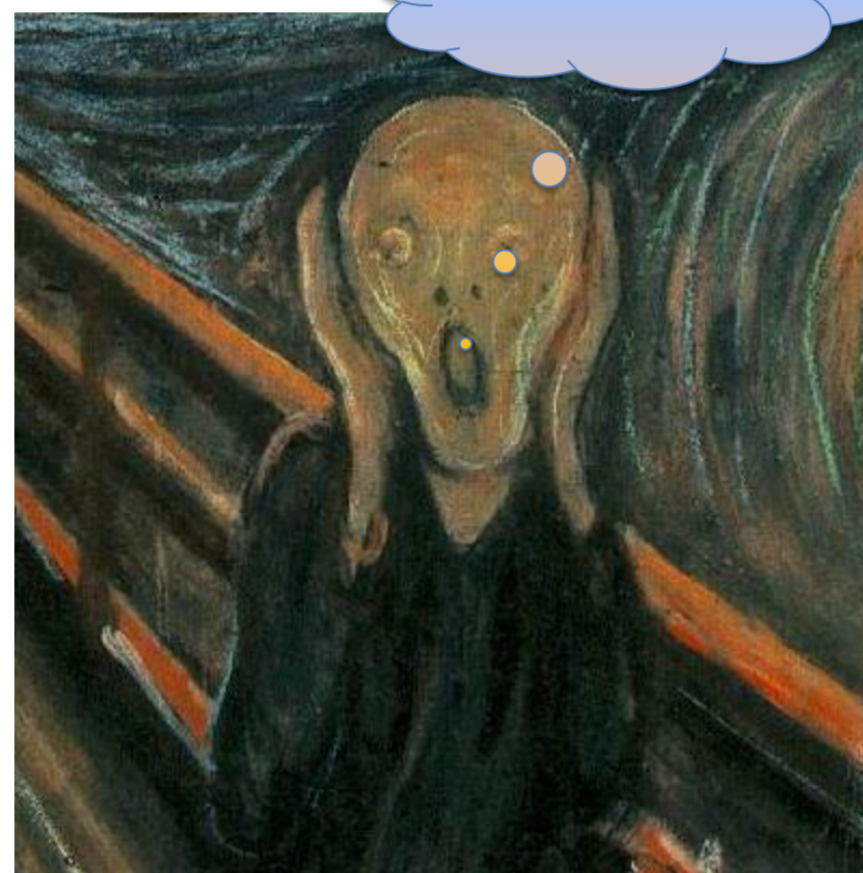
## CORAL "connection" and "session" for Oracle



## ORACLE DIRECT ACCESS VIA OCI



ORA-03113!!!



During the last two years the three experiments that make use of CORAL (ATLAS, CMS and LHCb) experienced a similar issue: an Oracle error appeared during the execution of some operations against the Oracle database, even though in different circumstances.

The Oracle error found is: **ORA-03113**

In some cases, this triggered an infinite loop and caused an application hang.

As reported by the Oracle site:

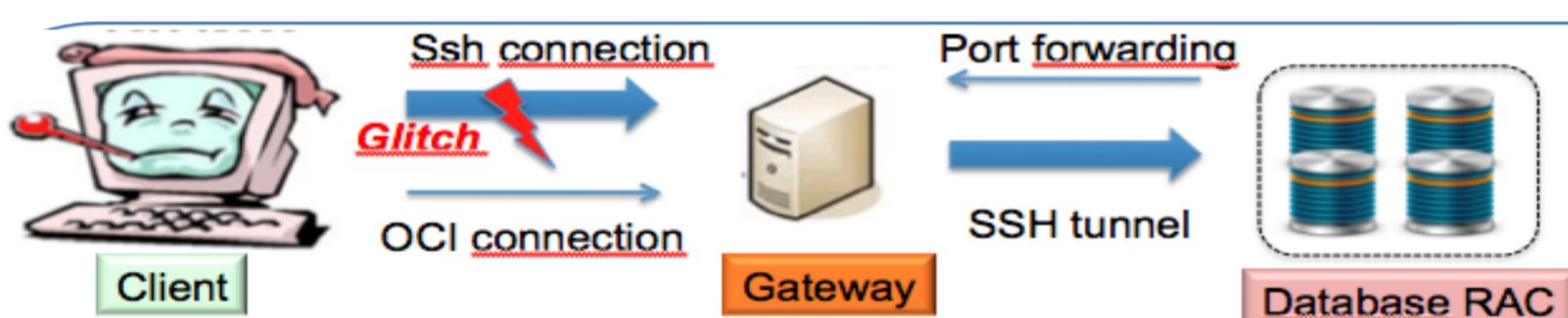
"ORA-03113: end-of-file on communication channel"

Cause: The connection between Client and Server process was broken.

Action: There was a communication error that requires further investigation."

The most likely cause of this issue **is an instability of the network**, leading to **a temporary connection break** between client and server.

## Network glitch error reports



To simulate a network glitch, the connection has been split into two branches:

- ✓ The listener port of the database server is forwarded to a gateway port via ssh.
- ✓ The client connects to the forwarded database listener on the gateway.

Our strategy consists in killing the ssh process to break the tunnel, simulating a glitch in the client connection to the database.

GLITCH

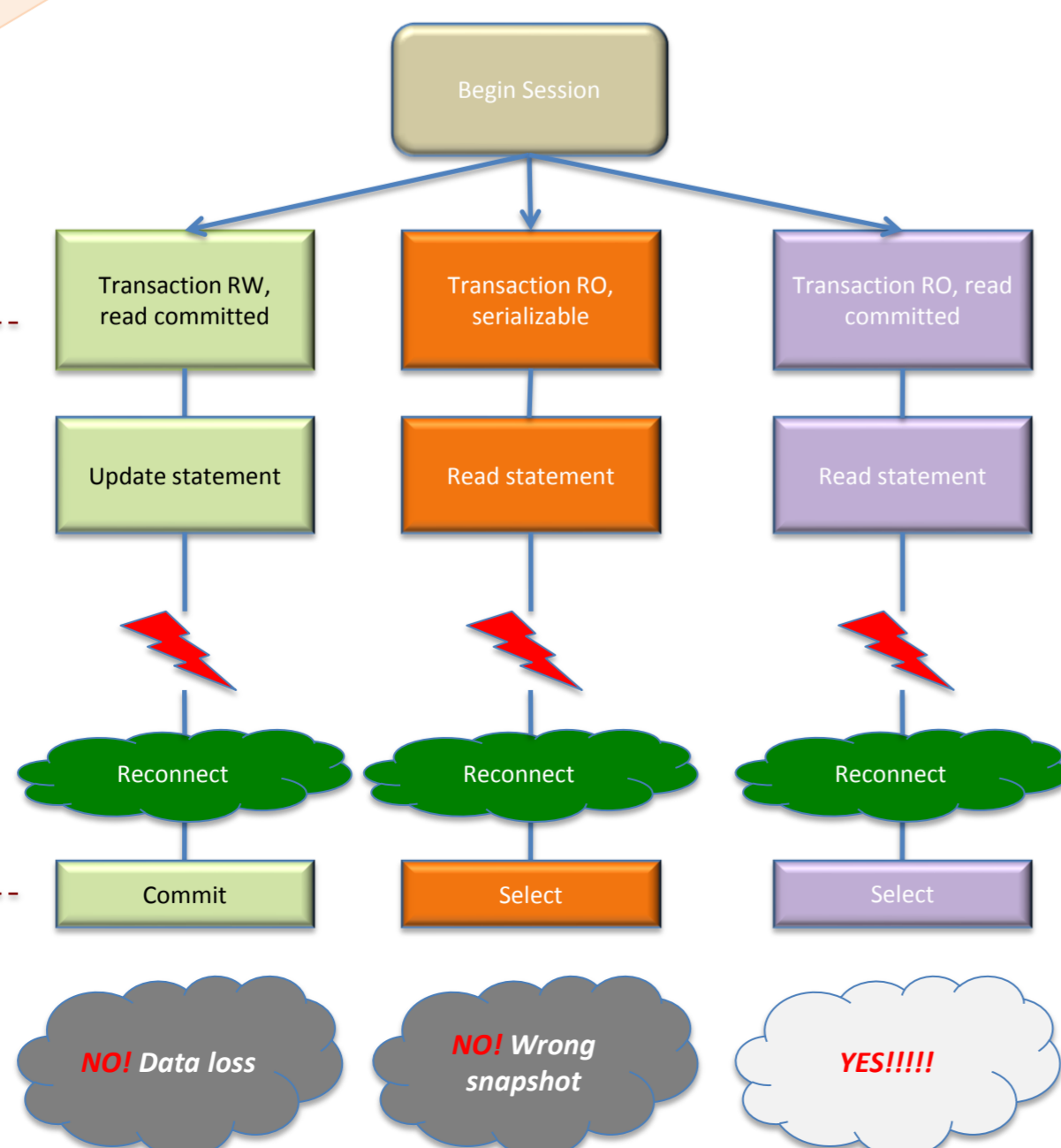
Network glitch test SET-UP

```
session = svc.connect(tunnelUrl, coral.access_ReadOnly )
transaction = session.transaction()
transaction.start(True)
schema = session.nominalSchema()
query = schema.tableHandle(tableName).newQuery()
cursor = query.execute()
```

- ✓ Create the physical connection and start a user session
- ✓ Retrieve a transaction handle
- ✓ Start a transaction
- ✓ Retrieve a schema handle
- ✓ Create a query
- ✓ Execute the query

Expected CORAL behaviour

When a network glitch occurs, the reaction of CORAL should depend on the type and status of the session and transaction.



## Preliminary bug fixes

During the analysis of the network glitch two other bugs have been identified, both related to the deletion order of the relevant CORAL objects (e.g. Session).

The first bug was that the destructors of many CORAL objects (e.g. Query) were using a Session already deleted. A **shared pointer** to the Session has been defined to keep it alive longer than all other objects.

The second bug was due to a wrong deletion order for OCI handles. Internally, most OCI handles (e.g. those for queries) use the service context handle when they are de-allocated. This was fixed by ensuring that **the service context is the last OCI handle to be de-allocated**.

## Network glitch fix

The validity of the connection and session is checked at the beginning of all crucial instructions (whenever the OCI service context handle is needed). A probe function has been implemented using the **OCIServerVersion function** to check the server accessibility.

If a network glitch is detected, CORAL reacts in the following way:

- ✓ If the transaction is not active yet, CORAL triggers a reconnection for any type of session;
- ✓ If a transaction is already active, CORAL triggers a reconnection only if the transaction has been started in RO read committed mode.

The reconnection procedure creates a new physical connection and starts a new user session.

**CORAL achieves this by refreshing all OCI handles without de-allocating and re-allocating them.**