# Operational QVT Incremental Update

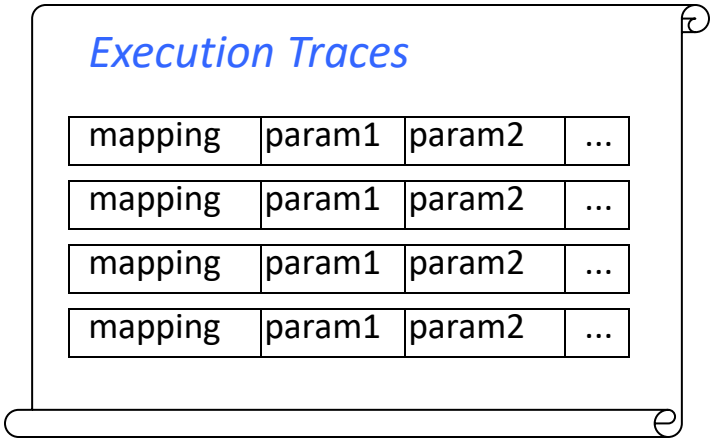
MOF 2.0 Query/View Transformation Specification:

**Incremental Update**

Once a relationship (a set of *trace instances)* has been established between models by executing a transformation, small changes to a source model may be propagated to a target model by re-executing the transformation in the context of the trace, causing only the relevant target model elements to be changed, without modifying the rest of the model.

# Normal execution flow

- Trace contains information about mapped objects
- Trace consists of trace records
- A trace record is created when a mapping is executed
- Trace records keep reference to the executed mapping and the mapping parameter values
- A trace record is created after the implicit instantiation section of the mapping is finished
- Trace prohibit duplicate execution with the same parameters
- Trace may be serialized after the transformation execution




```
mapping EPackage::myMapping() : Package {

        init {
        }

        population {
        }
        end {
        }
}
```

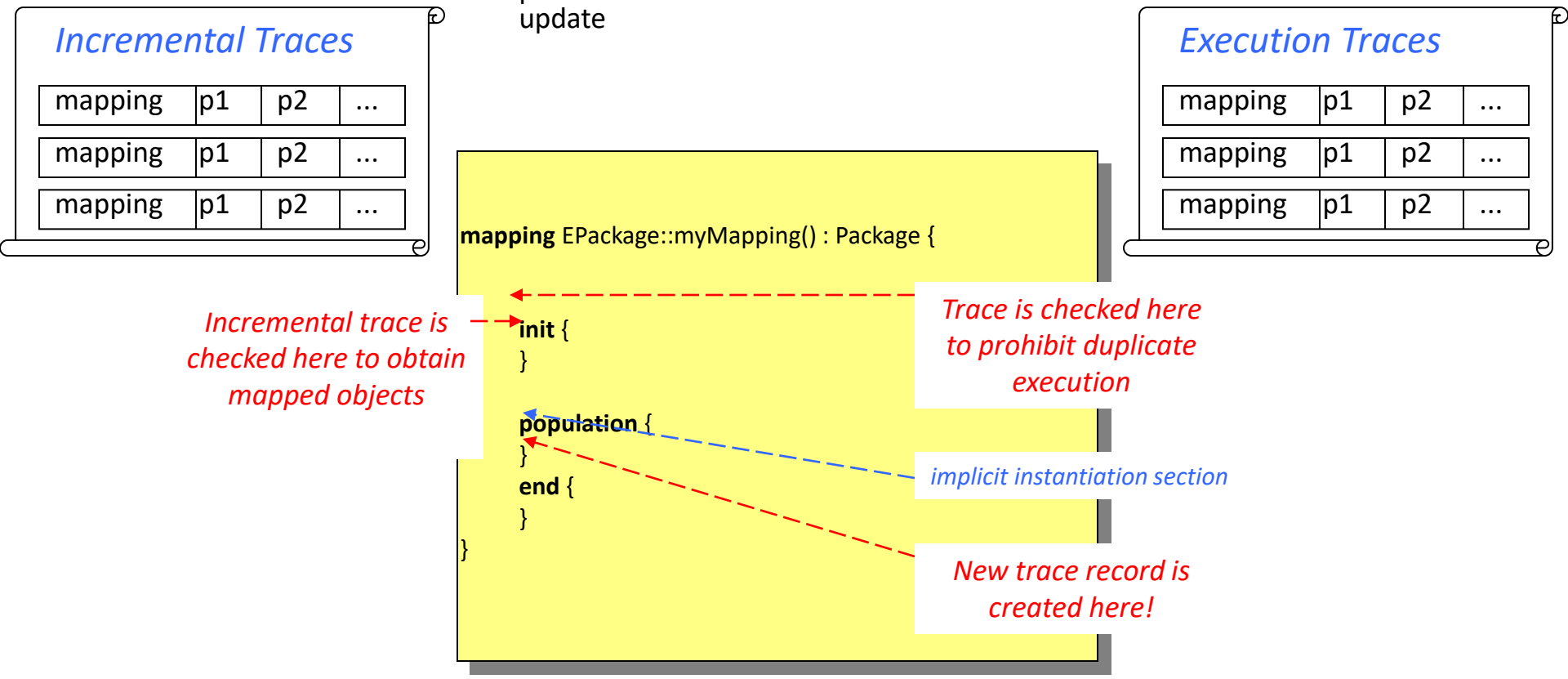
*Trace is checked here to prohibit duplicate execution*

*implicit instantiation section*

*New trace record is created here!*

# Incremental Update execution

- First execution of transformation creates traces
- Subsequent executions use traces from previous execution as a context for incremental update



# Transformation execution flow

- The whole transformation is executed (both in normal and in incremental execution modes)
  - it is because of imperative nature of QVTo transformations (in contrast to declarative transformations)
- EMF change notifications are generated only for actual changes in the output models
  - for single valued features this is optimized by EMF setters
  - for multi valued features this is optimized by means of ECollections.setEList(..)
- incremental update execution mode allows to stabilize 'xmi:id' for UML XMI format

# Transformation design for executing in incremental mode

- always use reset-assignment for multi valued features initialization (':=' operator)
- don't use conditional assignment for features

  Instead of

  ```
  if (true) {
     name := 'foo'
  }
  ```

  One should use

  ```
  name := if true then 'foo' else 'bar' endif;
  ```

- carefully use the assignment to 'result' variable in init{} sections
  consider using reuse facilities for mappings (inheritance, merge and disjunctions)