

Article

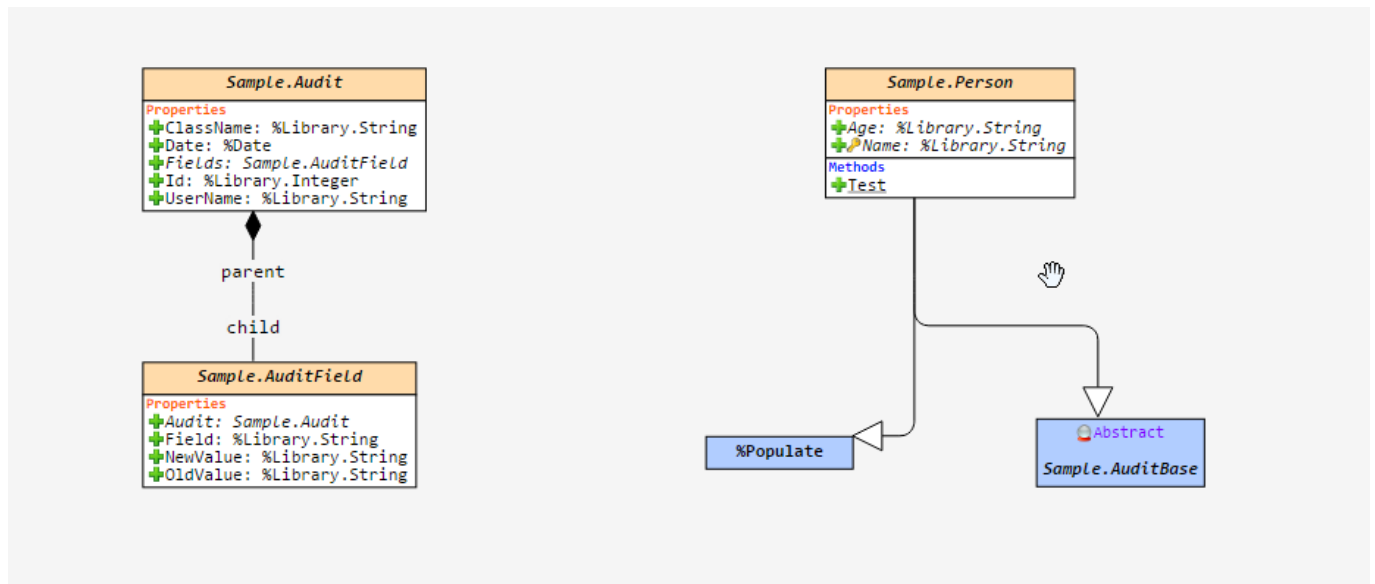
[Fabio Goncalves](#) · Apr 19, 2017 5m read

Tracking Data Changes - Audit Log - 2 of 2

In the [previous article](#), I have demonstrated a simple way to record data changes. At this time I have changed the "Audit Abstract Class" which is responsible for recording audit data and the data structure where the audit log will be recorded.

I have changed the data structure to a parent and child structure where there will be two tables to record the "transaction" and the "fields its values" changed on that transaction.

Take a look at the new data model:



Take a look at the code changed from "Audit Class":

```
Class Sample.AuditBase [ Abstract ]
{
```

```
Trigger SaveAuditAfter [ CodeMode = objectgenerator, Event = INSERT/UPDATE, Foreach =
row/object, Order = 99999, Time = AFTER ]
```

```
{
  #dim %compiledclass As %Dictionary.CompiledClass
  #dim tProperty As %Dictionary.CompiledProperty
  #dim tAudit As Sample.Audit

  Do %code.WriteLine($Char(9)_""; get username and ip adress")
  Do %code.WriteLine($Char(9)_"Set tSC = $$$OK")
  Do %code.WriteLine($Char(9)_"Set tUsername = $USERNAME")

  Set tKey = ""
  Set tProperty = %compiledclass.Properties.GetNext(.tKey)
  Set tClassName = %compiledclass.Name

  Do %code.WriteLine($Char(9)_"Try {")
  Do %code.WriteLine($Char(9,9)_""; Check if the operation is an update - %oper = UPDATE")
```

```

Do %code.WriteLine($Char(9,9)_"if %oper = ""UPDATE"" { ")

Do %code.WriteLine($Char(9,9,9)_"Set tAudit = ##class(Sample.Audit).%New()")
Do %code.WriteLine($Char(9,9,9)_"Set tAudit.Date = +$Horolog")
Do %code.WriteLine($Char(9,9,9)_"Set tAudit.UserName = tUsername")
Do %code.WriteLine($Char(9,9,9)_"Set tAudit.ClassName = "" _tClassName_ """)
Do %code.WriteLine($Char(9,9,9)_"Set tAudit.Id = {id}")
Do %code.WriteLine($Char(9,9,9)_"Set tSC = tAudit.%Save()")
do %code.WriteLine($Char(9,9,9)_"If $$$ISERR(tSC) $$$ThrowStatus(tSC)")
Do %code.WriteLine($Char(9,9,9)_"Set tAuditId = tAudit.%Id()")

While tKey '= "" {
set tColumnNbr = $Get($$$$EXTPROPSqlcolumnnumber($$$pEXT,%classname,tProperty.Name))
Set tColumnName = $Get($$$$EXTPROPSqlcolumnname($$$pEXT,%classname,tProperty.Name))

If tColumnNbr '= "" {

Do %code.WriteLine($Char(9,9,9)_"")
Do %code.WriteLine($Char(9,9,9)_"")
Do %code.WriteLine($Char(9,9,9)_"_tProperty.SqlFieldName)
Do %code.WriteLine($Char(9,9,9)_"if {" _tProperty.SqlFieldName _ "*"C"}")

Do %code.WriteLine($Char(9,9,9,9)_"Set tAuditField = ##class(Sample.AuditField).%New()")
Do %code.WriteLine($Char(9,9,9,9)_"Set tAuditField.Field = "" _tColumnName_ """)
Do %code.WriteLine($Char(9,9,9,9)_"Set tAuditField.OldValue = {" _tProperty.SqlFieldName_ "*O}")
Do %code.WriteLine($Char(9,9,9,9)_"Set tAuditField.NewValue = {" _tProperty.SqlFieldName_ "*N}")
Do %code.WriteLine($Char(9,9,9,9)_"Do tAuditField.AuditSetObjectId(tAuditId)")
Do %code.WriteLine($Char(9,9,9,9)_"Set tSC = tAuditField.%Save()")
do %code.WriteLine($Char(9,9,9,9)_"If $$$ISERR(tSC) $$$ThrowStatus(tSC)")
Do %code.WriteLine($Char(9,9,9)_"}")
}
Set tProperty = %compiledclass.Properties.GetNext(.tKey)
}

Do %code.WriteLine($Char(9,9)_"}")

Do %code.WriteLine($Char(9)_) Catch (tException) {}

Do %code.WriteLine($Char(9,9)_"Set %msg = tException.AsStatus()")
Do %code.WriteLine($Char(9,9)_"Set %ok = 0")
Do %code.WriteLine($Char(9)_)

Set %ok = 1
}

}

```

By changing data through the Test() classmethod, now you can see the "parent record" from the Audit Class (Sample.Audit) and the "children fields" changed from "Audit Field" class. (Sample.AuditField).

```

d ##class(Sample.Person).Test(1)
INSERT INTO Sample.Person (Name, Age) VALUES ('TEST PARENT-CHILD', '01')
SQLCODE: 0

```

```

ID Age Name
1 01 TEST PARENT-CHILD

```

1 Rows(s) Affected

UPDATE Sample.Person SET Name = 'INTERSYSTEMS DEVELOPER COMMUNITY', Age = '100' WHERE Name = 'TEST PARENT-CHILD'
SQLCODE:0

ID Age Name

1 100 INTERSYSTEMS DEVELOPER COMMUNITY

1 Rows(s) Affected

Audit classes:

The screenshot shows the InterSystems SQL interface. The left sidebar displays a tree view with 'Sample.Audit' selected. The main window shows the 'Execute Query' tab with the following SQL statement:

```
SELECT
ID1, ClassName, "Date", Id, UserName
FROM Sample.Audit
```

The results table shows 1 row(s) affected:

ID1	ClassName	Date	Id	UserName
1	Sample Person	19/04/2017	1	_SYSTEM

The screenshot shows the InterSystems SQL interface. The left sidebar displays a tree view with 'Sample.AuditField' selected. The main window shows the 'Execute Query' tab with the following SQL statement:

```
SELECT
Audit, ID, Field, NewValue, OldValue, childsub
FROM Sample.AuditField
```

The results table shows 2 row(s) affected:

Audit	ID	Field	NewValue	OldValue	childsub
1	1	Age	100	01	1
1	1	Name	INTERSYSTEMS DEVELOPER COMMUNITY	TEST PARENT-CHILD	2

Note that the Sample.AuditField records have references to the Sample.Audit class through Audit field = 1. You can query data by using the relationship from both classes as follows:

The screenshot shows the InterSystems SQL interface. The left sidebar displays a tree view with 'Sample.AuditField' selected. The main window shows the 'Execute Query' tab with the following SQL statement:

```
SELECT Audit->ClassName, Audit->Id, Audit->UserName, NewValue, OldValue
FROM Sample.AuditField
```

The results table shows 2 row(s) affected:

ClassName	Id	UserName	NewValue	OldValue
Sample Person	1	_SYSTEM	100	01
Sample Person	1	_SYSTEM	INTERSYSTEMS DEVELOPER COMMUNITY	TEST PARENT-CHILD

That's it. As a result we have a different audit log data structure.

[#Object Data Model](#) [#ObjectScript](#) [#Caché](#)

Source URL: <https://community.intersystems.com/post/tracking-data-changes-audit-log-2-2>