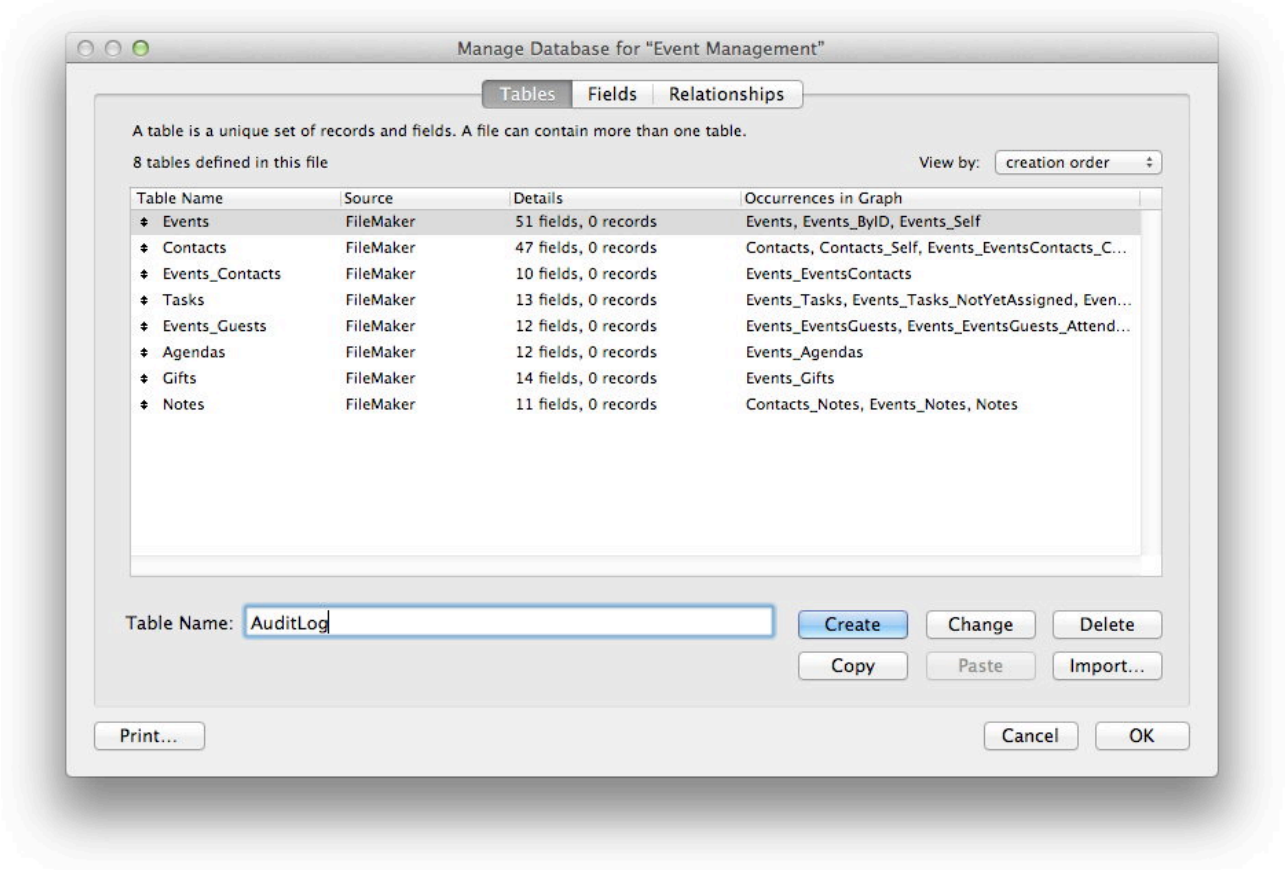
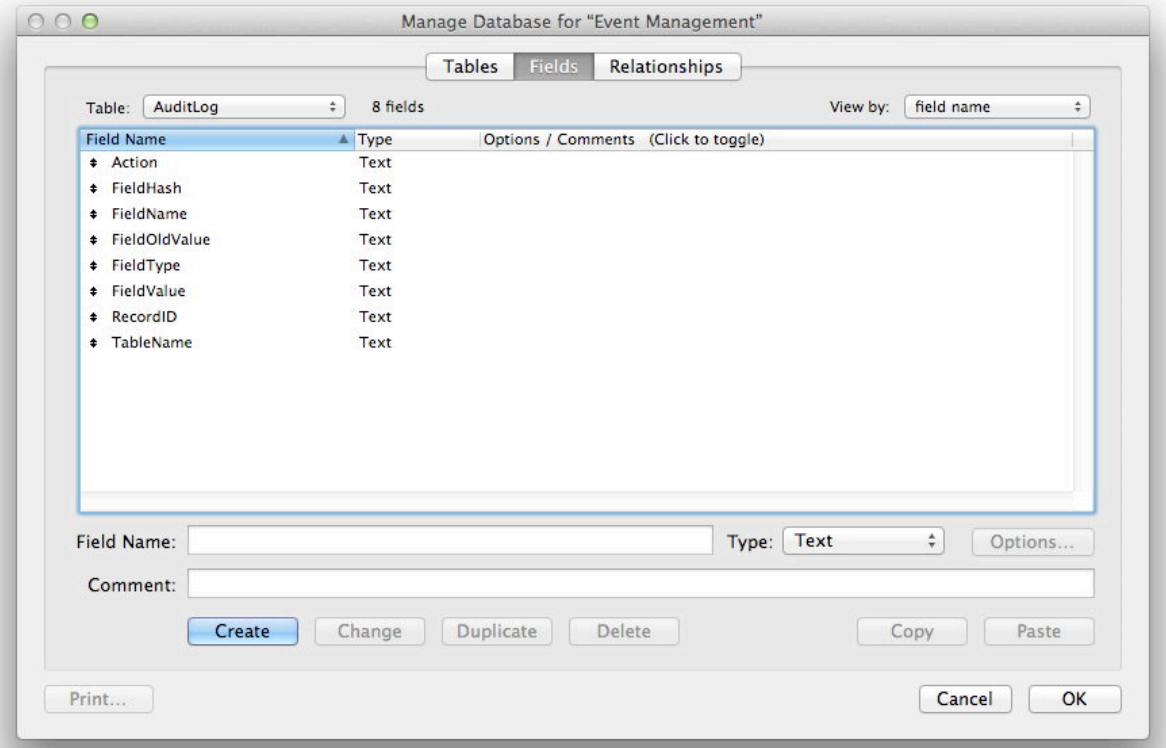
Introducing Audit Plugin Functions for Filemaker

Sometimes people want to know who edited which record when and what values changed. All this questions can be solved with an audit log and our plugins. The idea is simple: Whenever someone changes something, we write a records for those changes to the AuditLog. Later you can check the log or have a script restore the changes.

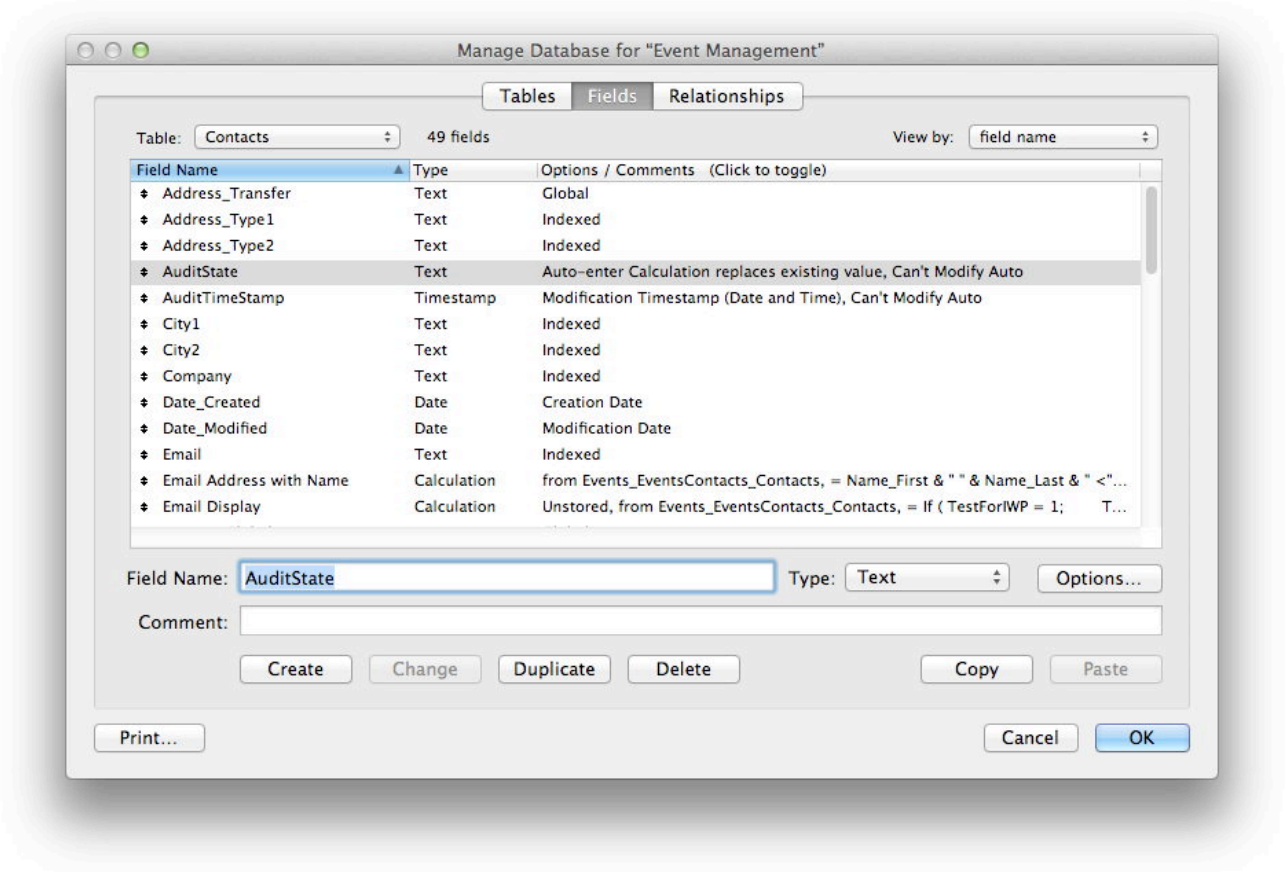
The MBS Filemaker Plugin 2.8 introduces a few new Audit functions.



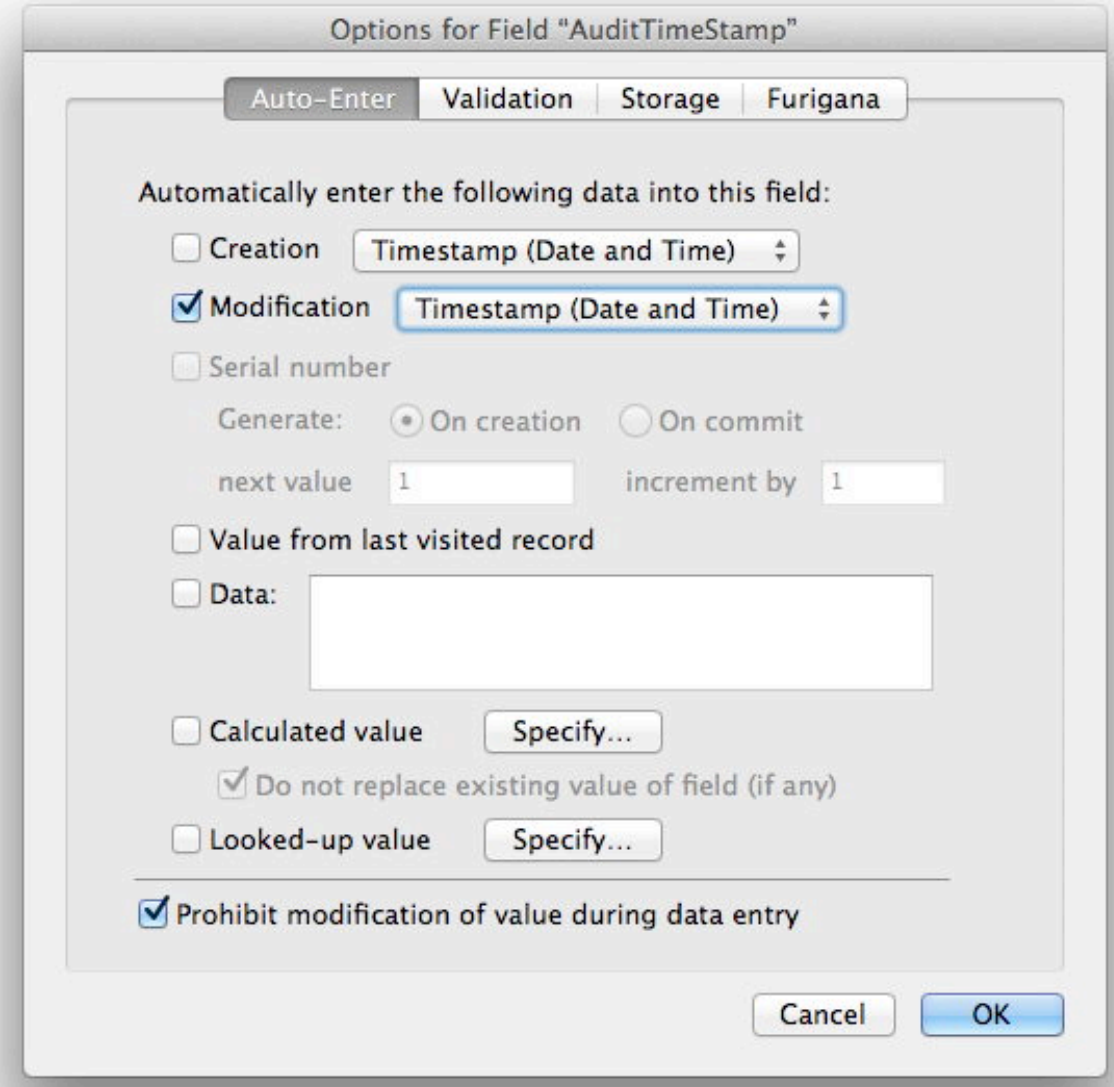
To show you how it works, we add Audit Logging to one of the starter solutions, the Event Management database. First we go and create a new table named AuditLog. This table can be in a different database file if needed as long as the plugin can find a table with that name. But in all cases, we need you to have a layout for the AuditLog table. It's not required to be user visible, but the plugin needs to find it.



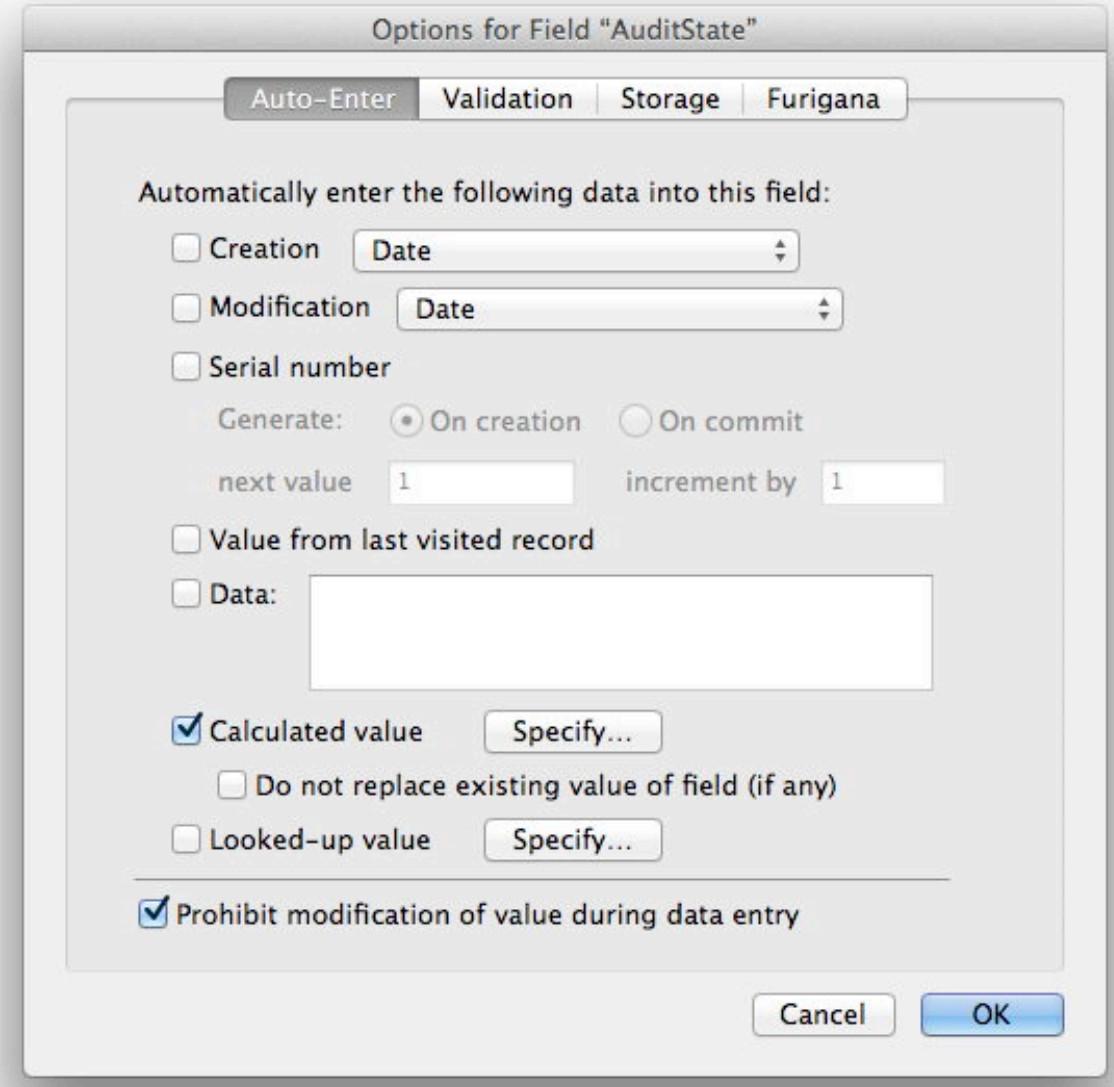
Next we create a couple of fields. Required for the plugin are FieldName, FieldHash, TableName, RecordID. So we can store which table and which field changed. RecordID is the unique ID of the record and FieldHash stores a hash value for the content of the field. Optionally you can add more fields: FieldValue, FieldOldValue, FieldType, UserName, IP, CurrentTimestamp, TimeStamp, CurrentTime, CurrentDate, Action, CurrentHostTimeStamp, PrivilegeSetName, AccountName, LayoutNumber, ApplicationVersion, FileName, HostApplicationVersion, HostName, HostIPAddress, LayoutName, PageNumber, LayoutTableName, TableID, FieldID and WindowName. In this example we added FieldValue and FieldOldValue, so we know the new and old values for the change. The action field stores what happened and FieldType can tell us what data type we have for the value.
You can later add more fields if you like. The plugin dynamically detects them and fills them with values. Like if you need to know the IP of the user, simply add a new IP field and all new log entries record the IP addresses.



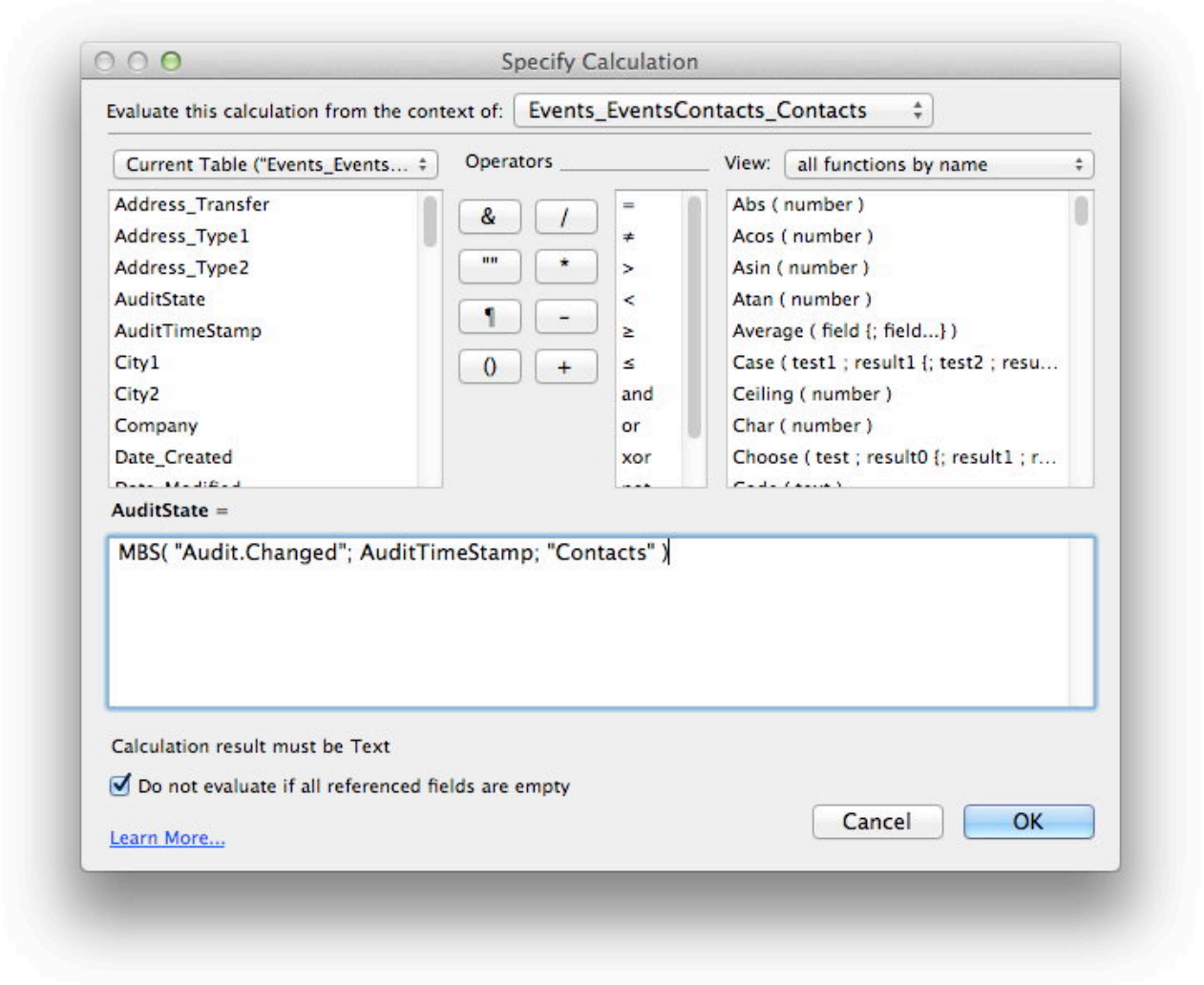
Now we can check all the tables in the database. For each we create two fields. First AuditTimeStamp with the timestamp when record was last modified. And second the AuditState calculated field which calls our plugin. The fields do not need to have exact this names. But if you don't use the default names, you have to change them everywhere and inform the plugin.



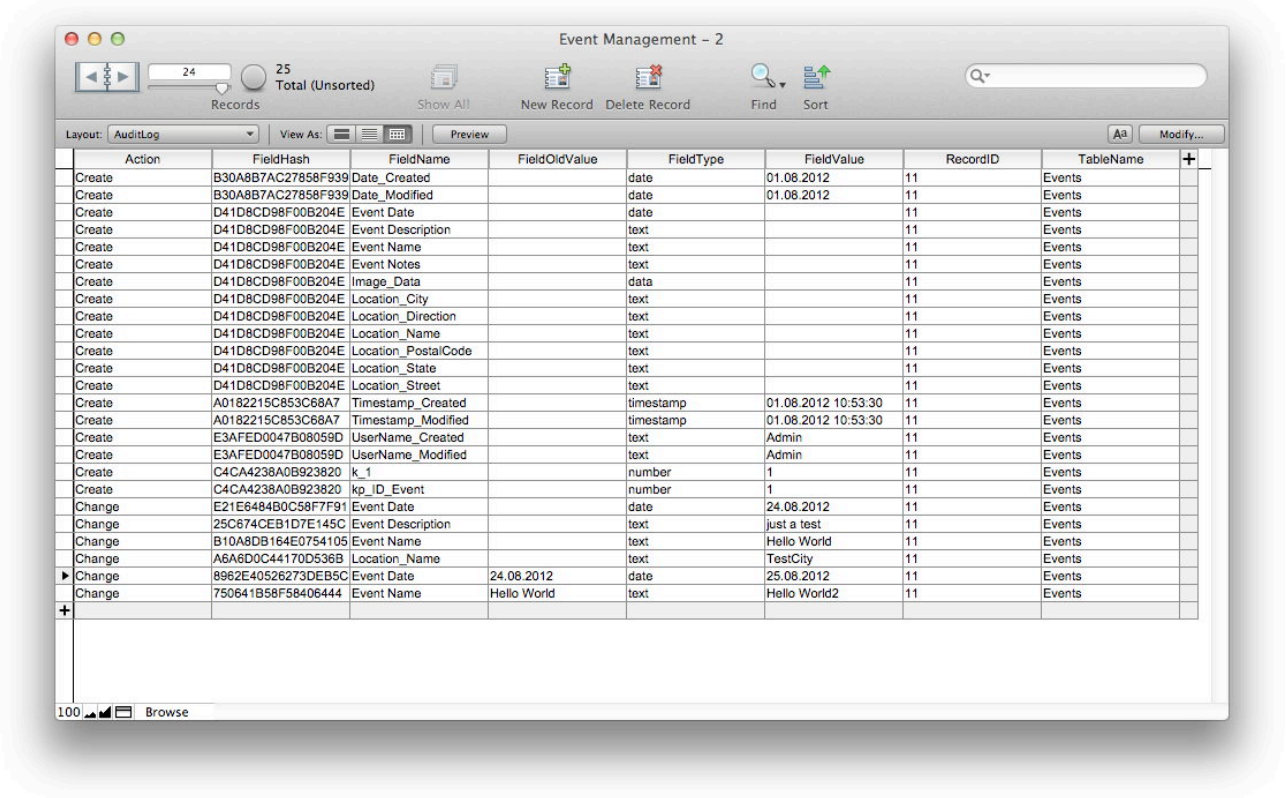
Here you see the definition for the time stamp field. We check the checkbox to store here the record modification timestamp. Filemaker will update this field automatically every time the record changes.
We typically do not allow the user to edit the Audit fields.



We define the AuditState field. Simply a text field which is calculated. Also make sure the checkbox "Do not replace existing value of field (if any)" is unchecked.
Here we also declare that user should not edit the field.



In the calculation for the AuditState field, we call the plugin: MBS("Audit.Changed"; AuditTimeStamp; "Contacts"). As you see we call the Audit.Changed function and pass the time stamp field from above. Third parameter is the name of the table. Filemaker simply does not tell the plugin what name the current table has, so we need to pass it here. If the table contains fields which you don't want to be logged or you have given AuditState/AuditTimeStamp fields a different name, you should pass them here as additional parameters. So for example you can call MBS("Audit.Changed"; AuditTimeStamp; "Contacts"; "myAuditState"; "myTimeStamp"; "internalField"). This way the plugin will not log those three fields. Unstored calculations and global fields are never logged. With the function Audit.SetIgnoredFieldNames you can globally define which fields you want to ignore always.



Action	FieldHash	FieldName	FieldOldValue	FieldType	FieldValue	RecordID	TableName
Create	B30A8B7AC27858F939	Date_Created		date	01.08.2012	11	Events
Create	B30A8B7AC27858F939	Date_Modified		date	01.08.2012	11	Events
Create	D41D8CD98F00B204E	Event Date		date		11	Events
Create	D41D8CD98F00B204E	Event Description		text		11	Events
Create	D41D8CD98F00B204E	Event Name		text		11	Events
Create	D41D8CD98F00B204E	Event Notes		text		11	Events
Create	D41D8CD98F00B204E	Image_Data		data		11	Events
Create	D41D8CD98F00B204E	Location_City		text		11	Events
Create	D41D8CD98F00B204E	Location_Direction		text		11	Events
Create	D41D8CD98F00B204E	Location_Name		text		11	Events
Create	D41D8CD98F00B204E	Location_PostalCode		text		11	Events
Create	D41D8CD98F00B204E	Location_State		text		11	Events
Create	D41D8CD98F00B204E	Location_Street		text		11	Events
Create	A0182215C853C68A7	Timestamp_Created		timestamp	01.08.2012 10:53:30	11	Events
Create	A0182215C853C68A7	Timestamp_Modified		timestamp	01.08.2012 10:53:30	11	Events
Create	E3AFED0047B08059D	UserName_Created		text	Admin	11	Events
Create	E3AFED0047B08059D	UserName_Modified		text	Admin	11	Events
Create	C4CA4238A0B923820	k_1		number	1	11	Events
Create	C4CA4238A0B923820	kp_ID_Event		number	1	11	Events
Change	E21E6484B0C58F7F91	Event Date		date	24.08.2012	11	Events
Change	25C674CEB1D7E145C	Event Description		text	just a test	11	Events
Change	B10A8DB164E0754105	Event Name		text	Hello World	11	Events
Change	A6A6D0C44170D536B	Location_Name		text	TestCity	11	Events
Change	8962E40526273DEB5C	Event Date	24.08.2012	date	25.08.2012	11	Events
Change	750641B58F58406444	Event Name	Hello World	text	Hello World2	11	Events

When all tables have been prepared, you can use the database. As you see, the first time the plugin sees a record, it writes log entries with "Create" as action. Next time you touch the record, you see "Change" entries for all the changes you made.
You can later add more fields to be logged like UserName or WindowName.

With Audit.Delete function you can also log deletion of records. But that is a topic for another blog entry.

If you have questions, please do not hesitate to contact us.