

Design Tip #74 Compliance-Enabled Data Warehouses

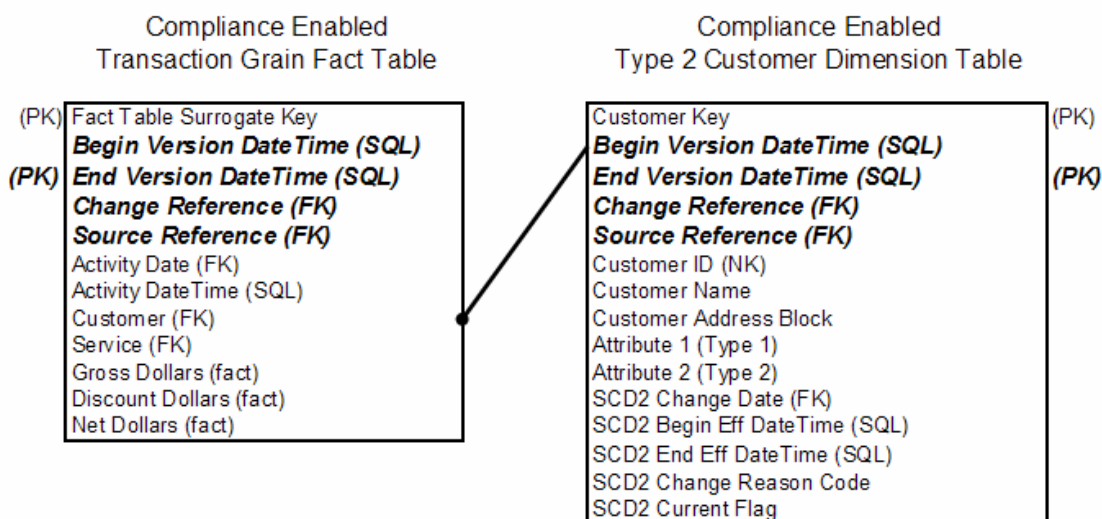
By Ralph Kimball

I often describe compliance in the data warehouse as “maintaining the chain of custody” of the data. In the same way a police department must carefully maintain the chain of custody of evidence in order to argue that the evidence has not been changed or tampered with, the data warehouse must also carefully guard the compliance-sensitive data entrusted to it from the moment it arrives. Furthermore the data warehouse must always be able to show the exact condition and content of such data at any point in time that it may have been under the control of the data warehouse. Finally when the suspicious auditor is looking over your shoulder, you need to link back to an archived and time stamped version of the data as it was originally received, which you have stored remotely with a trusted third party. If the data warehouse is prepared to meet all these compliance requirements, then the stress of being audited by a hostile government agency or lawyer armed with a subpoena should be greatly reduced.

The big impact of these compliance requirements on the data warehouse can be expressed in simple dimensional modeling terms. Type 1 and Type 3 changes are dead. Long live Type 2. In other words, all changes become inserts. No more deletes or over-writes.

In plain English, the compliance requirements mean that you cannot actually change any data, for any reason. If data must be altered, then a new version of the altered records must be inserted into the database. Each record in each table therefore must have a begin-timestamp and an end-timestamp that accurately represents the span of time when that record was the “current truth.”

The following figure shows a compliance-enabled fact table connected to a compliance-enabled dimension table. The fields shown in bold-italics in each table are the extra fields needed for compliance tracking. The fact table and the dimension table are administered similarly.



The **Begin Version DateTime** and **End Version DateTime** fields describe the span of time when the record in question was the “current truth.” The **Change Reference** field is a foreign key pointing to a Change Reference dimension (not shown) that describes the change status of that record. The **Source Reference** field is a foreign key pointing to a Source Reference dimension (not shown) that shows the off-site trusted third party location where the hash-encoded and time stamped version of this record has been stored since the moment when it was created.

The first time that records are loaded into either of the tables, the **Begin Version DateTime** is set to the time of the load, and the **End Version DateTime** is set to an arbitrary date far in the future, such as midnight, December 31, 9999. The Change Reference would describe the change status as “initial load.” If and when a Type 1 correction needs to be made to either facts in the fact table or Type 1 or 3 attributes in the dimension table, a new record is inserted into the respective table with the same surrogate key, but a new pair of **Version DateTimes**. Thus, the true primary key of the tables has become the combination of the original primary key field and the **End Version DateTime**, as shown in the figure. When the new record is inserted, the prior most recent **End Version DateTime** field must be changed to this new load **DateTime**. This two-step dance of adjusting the **Begin** and **End DateTime** stamps is familiar to data warehouse architects from the normal Type 2 dimension processing.

The presence of existing Type 2 attribute processing in the dimension does not change anything. Only Type 1 and Type 3 updates invoke the special steps described in this Design Tip. Type 2 processing proceeds as it always has, by introducing new dimension member records with new surrogate primary keys, and by administering the familiar SCD Type 2 metadata fields shown in the bottom portion of the dimension in the figure.

If any of you are uncomfortable with the seeming introduction of a two part key in the dimension table (a violation of the standard dimensional design that insists on a single field surrogate key), then think about it in the following way. Although technically one must view the tables as now having two-part primary keys, think about this schema as still behaving like typical one-part key designs ***after the version has been constrained***. That’s the point of this Tip. First, you choose a version ***as of*** a certain date, then you run your typical queries with the same old keys as always.

Before concluding this Design Tip, let’s not overlook the BIG benefit of this approach! When the hostile government agency or lawyer armed with a subpoena demands to see what has happened to the data, your tables are fully instrumented to comply. If you must reveal the exact status of the data as of, say, January 1, 2004, then you just constrain this date to be between the **Begin** and **End Version DateTimes** of all the tables. Presto – the database reverts to that moment in history. And, of course, any subsequent changes made to that data are fully explained by the **Change Reference**. Finally, you can prove that the evidence (oops, I mean data) hasn’t been tampered with by using the **Source Reference**.

Good luck with your compliance.