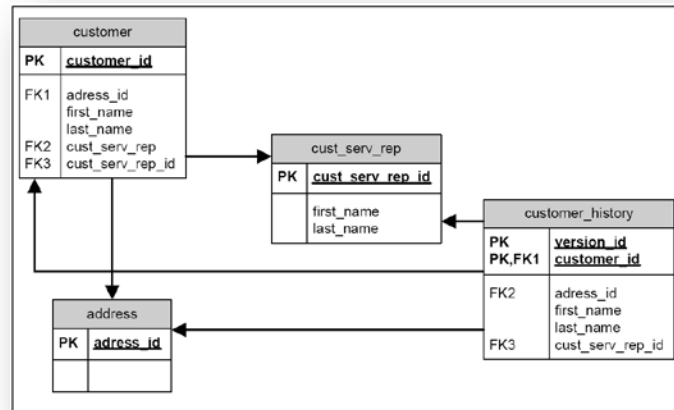


## Audit logging

With laws requiring the storage of historical data over several years ER-designs become quite large. Typically the table to be versioned is duplicated in structure. For every update or delete statement the old data set is then copied to the history table.

The exact table layout depends on the level of detail that is to be versioned. Figure 1 shows a complete table versioning of the customer table. Additionally to the *customer\_id* a *version\_id* is assigned to every dataset in the history table. Once a dataset changes the history is filled accordingly. This method not only explodes the table design but also adds error prone code to the applications. The described method above is not the only way. Data versioning could also be done via triggers and different looking table designs.



All solutions described above manage the versioning manually. Either by application code or database functions. With MySQLs pluggable storage engine architecture the versioning of data can be automated. The database programmer simply uses the table as usual. Versioning of the data is taken over by the storage engine.

A simple *SELECT \* FROM customer* would return all versions of the customers data. By submitting either a query selecting a specific version of a data set or selecting the maximum version of a dataset the correct version is returned. In the final release the engine will be able to version the data in three different ways:

1. The first option is to version the complete data set similar to the table design as Figure 1 shows. This option should be used for tables containing fields that underlie a constant change. But this introduces a larger disk footprint. Selects on a specific version are with this first version a lot faster since the data is available without any further calculation.
2. Another option will be to only store the difference of the new data set and the old data set. This ensures a small disk footprint and fast Insert and Update statements. In comparison to the first option the retrieval of historical data is slower since the data must be merged back together.
3. The simplest version will be available in alpha release. It will store the data next to the current data in the same table space. This will ensure fast Insert and Update statements but slow selects statements down due to high I/O throughput.