**Slowly Changing Dimension (SCD):**

A Slowly Changing Dimension (SCD) is a dimension that stores and manages both current and historical data over time in a data warehouse.

A slowly changing dimension (SCD) is a dimension that is able to handle data attributes which change over time. For example:

A customer dimension may hold attributes such as name, address, and phone number.

Over time, a customer's details may change (e.g. move addresses, change phone number, etc).

**SCD Types:**  
#1 SCD Type 0 — Dimension is never updated

#2 SCD Type 1 — Dimension is overwritten

#3 SCD Type 2 — Maintain all the old records for the dimension by versioning the row

#4 SCD Type 3 — Maintain an old attribute but in the same row (by adding a previous value column)

#5 SCD Type 4 — Maintain current record and older record in two different tables

#6 SCD Type 6 — Type 6 is a hybrid of 1,2,3 i.e. 1+2+3 = 6

**#1 SCD Type 0 — Dimension is never updated**

Simply means, as a business, we do not want to update the Currency column even if William pays using USD or any other currency. It could also mean that nothing other than GBP is allowed for transactions, and hence this table will always have GBP till eternity.

The basic premise of Type 0 is that the changes in data do not impact the dimension.



This type of SCD is generally used for reference data where classifiers such as currency or country names, codes etc., is used. There is rarely a need to delete or update the underlying data.

**#2 SCD Type 1 — Dimension is overwritten**

In this case, as William starts paying using USD, GBP as a value is overwritten. This is perfectly acceptable, especially if there is no analytical purpose of retaining GBP and USD simultaneously.

Before:



After:



The question to ask the business is: what is the benefit of knowing the currency customer used previously? Are you treating customers paying using USD different to those paying using GBP? If so, it may be a good idea to use Type 2.

**#3 SCD Type 2 — Maintain all the old records for the dimension by versioning the row**

This is the most common type of SCD in data warehousing for large organisations.

Instead of overwriting, a row is added with the latest information below.

Before:



After:



However, this has caused a complication where a singular customer has two associated currencies. Due to this, there is a need for additional columns to help with the reporting.

We need a unique key to recognise each record, start and end dates to signify when a new record was added, and a flag to show which one is the current record. This is generally built into the data pipelines as the records are inserted.

The tables will look like this.



A surrogate key is an unnatural key added to help differentiate the two records.

Date\_From helps explain when these records were added to the datawarehouse

Date\_To help describe how long the records were valid until. High-End Date is used for the latest open records.

Delete\_Flag could also be a Current Record flag. I prefer Delete\_Flag as it acts as two things:

1. Helps find the latest record rather than having to work it out using dates

2. Helps to purge the data once the agreed retention period is over; simply filter by Delete\_Flag = 1 and then purge.

**#4 SCD Type 3 — Maintain an old attribute but in the same row (by adding a previous value column)**

Like Type 2, except it is column versioning rather than row.

Before:



After:



This is great if all you need is an immediate previous value, and you do not want to introduce the concept of surrogate keys and the latest record flag. However, if you would like to maintain each historical event, including if William starts paying in EUR, you must use Type 2. Otherwise, you will only have EUR and USD in the future and lose the event when William was once paying in GBP.

**#5 SCD Type 4 — Maintain current record and older record in two different tables**

This is a combination of Type 1 and Type 2 by storing the historic dimension information in different tables.

Before:



After:

**Table\_1 (just like Type 1)**



**Table\_2 (history store table)**



Unlike Type 2, we need not maintain the end date or record flag in Table\_2 as the latest record is already stored in Table\_1.