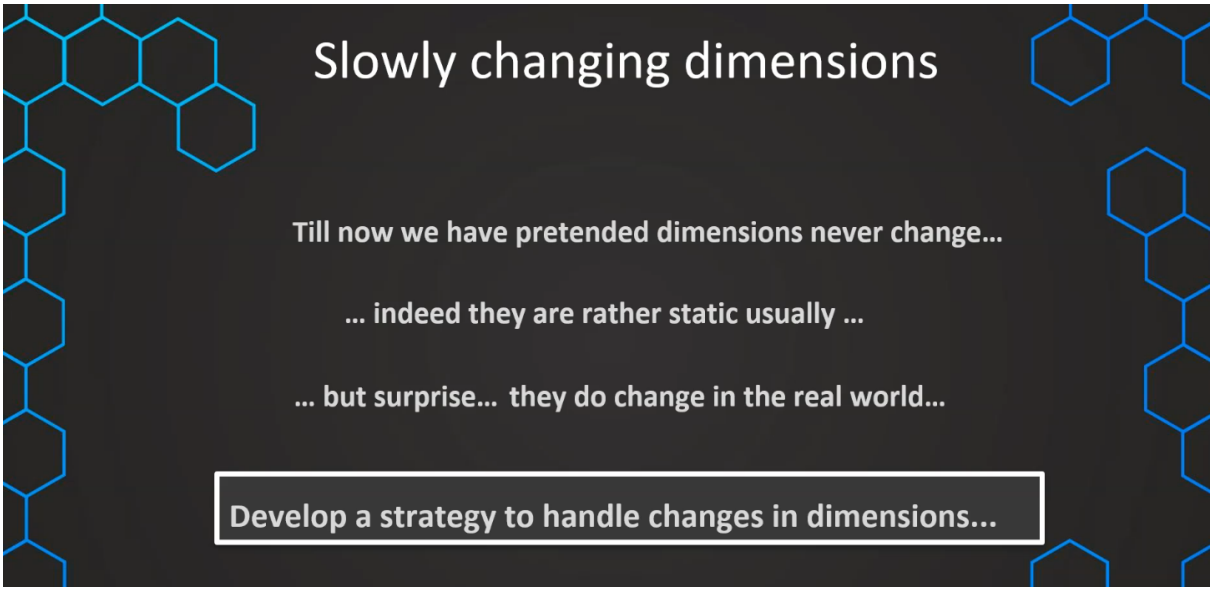
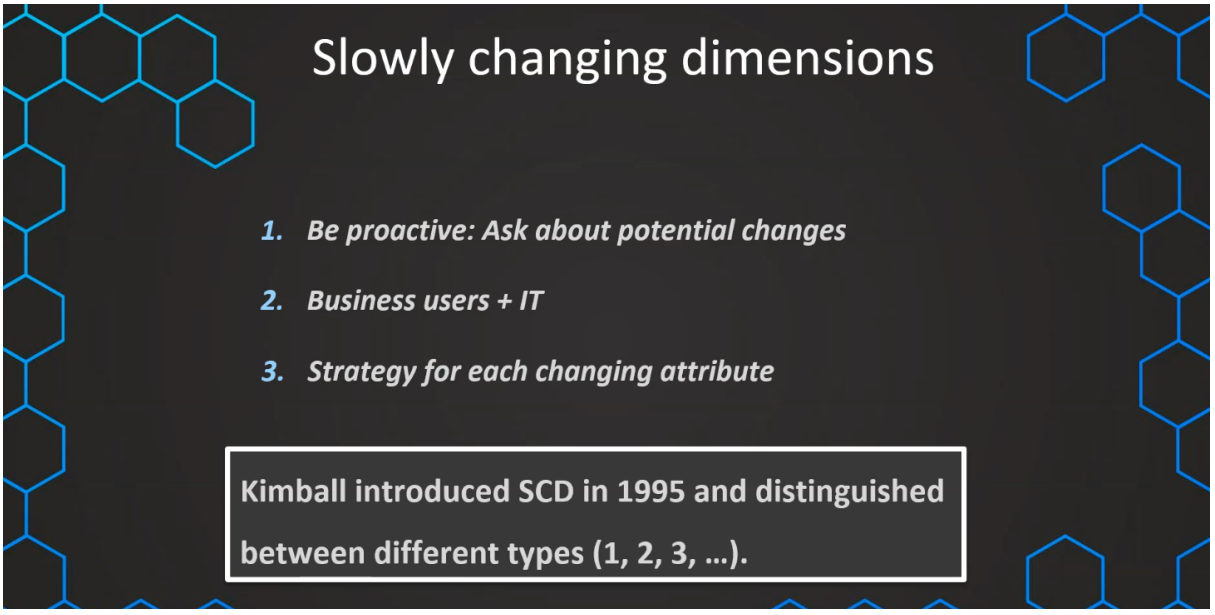
08. Slowly changing dimensions

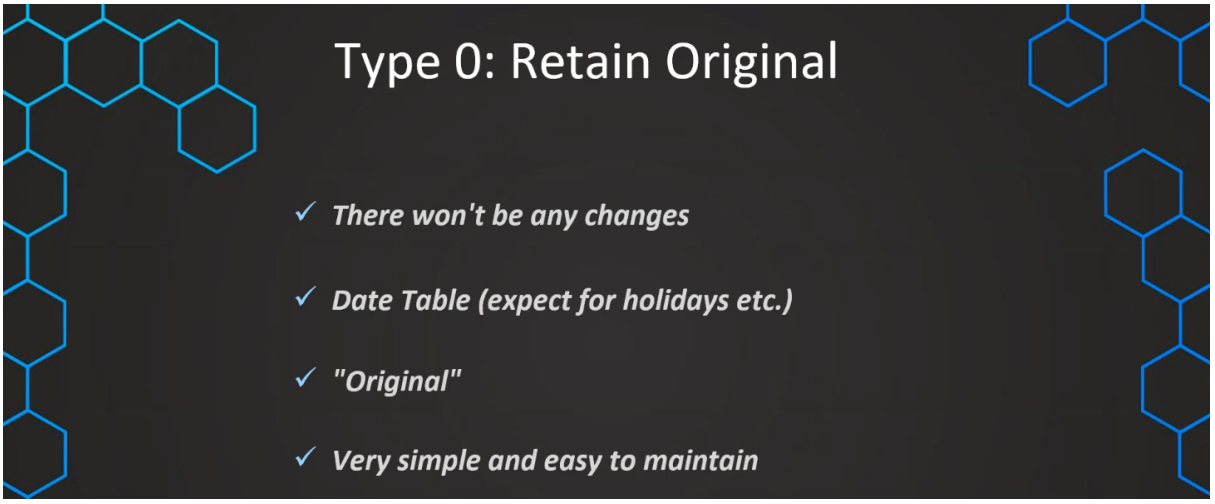


- This is a term that is quite popular in the world of data warehousing, and therefore we need to talk about why this is important for us and actually how we can handle changes in our dimensions.
- Because up until now, we've just pretended, in a way, that there would not be any changes, any modifications in our dimensions And indeed, the dimensions compared to Facts are rather static.
- So oftentimes we don't really expect many changes there or any changes at all But, surprise, in the real world of course, there are some changes in our dimensions as well.
- So some attributes might change and therefore we need to also have a strategy in place to handle those changes in our dimensions.
- quick note on how we should go about that in general, because oftentimes the business users are not expecting that there are any changes in our dimensions. And therefore don't expect that they tell you that there are changes and you need to handle them.



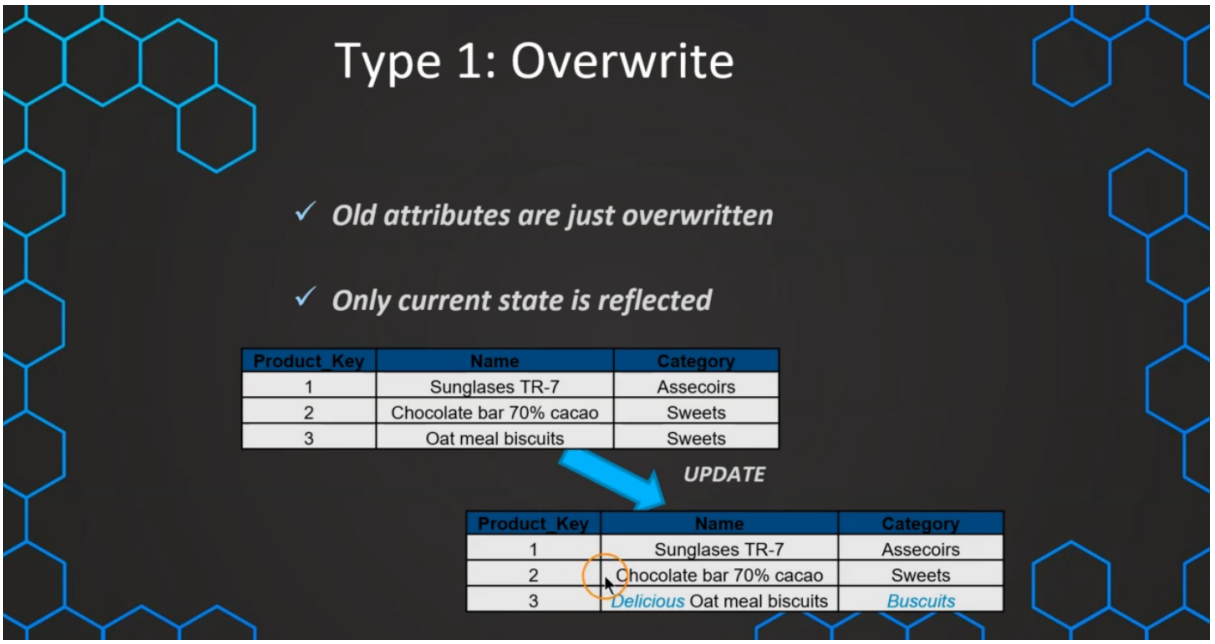
- But just be proactive and ask about potential changes. And also, again, not only ask the business users, but also the people that are maybe in the IT. And therefore, sometimes business users don't know about these changes, they are not so aware about them.
- And therefore bring together, ideally, the business users and some IT responsibles for these source data systems. And therefore, in that case, if we have that, we can now develop a strategy together with these people for these changing dimensions.
- So some attributes in our dimensions might change and that's why for each of these attributes we need to develop an own strategy. And now, depending on the situation and the requirements, we can distinguish between different types of slowly changing dimensions.
- Those different types of slowly changing dimensions were first introduced by Kimball in 1995. And we will also use these types of dimensions because they are probably the most famous ones and therefore we want to focus on them.
- And also note that the term SCD, it's just abbreviation of slowly changing dimensions, it's the most common expression. And therefore, if you see this abbreviation, this is just referring to the slowly changing dimensions.

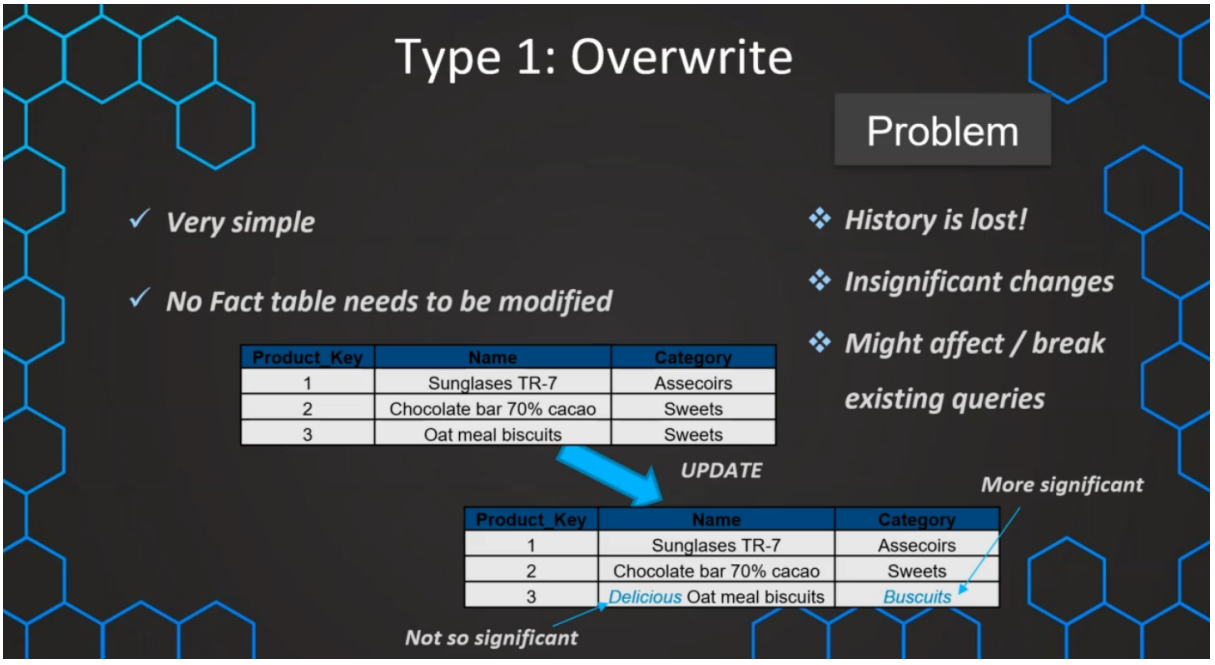
Type 0 - Retain original



- the first type of slowly changing dimensions in which we only retain the original data.
- This is applicable if there are no changes in our dimensions. So we really need to make sure that there are no changes occurring in our dimension. And in that case, of course, we don't need to do anything.
- We don't need to implement any strategy. We can just leave the dimensions as they are.
- This is usually applicable for **date** tables except there can be some attributes in this date table, for example, company holidays that might be subject to changes as well. But in general, the date table is really a static table that is not really having any changes.
- So this is a common type. Also, there can be many attributes that are just called or labeled as original. So for example, the original product name or something that we just know. This is just kept as it is and we will not have any changes in that.
- So of course, if we have such a situation, such an attribute in our dimension table, this is the most easiest option because we don't need to make any changes.
- We can just load these attributes in our table and just not make any changes in here. And therefore, if we happen to have such a situation this is great because there's no additional work and no additional thoughts necessary.
- But just be aware that you are really sure that there are no changes or at least, those changes don't need to be reflected in our dimension table.

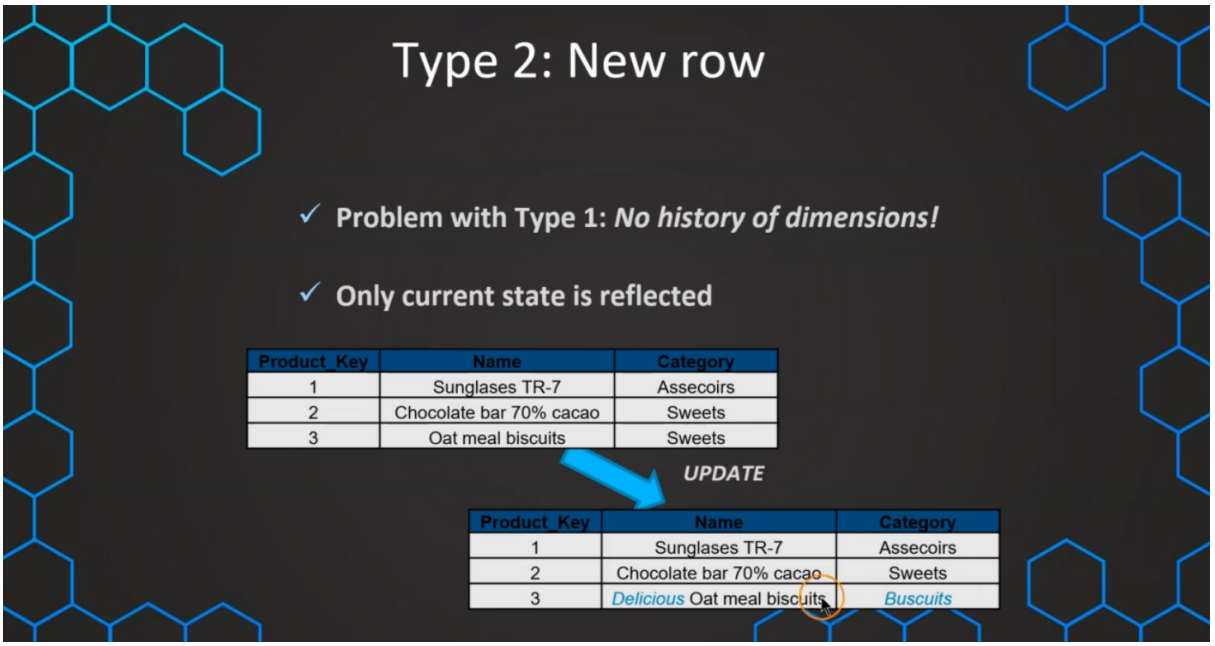
Type 1: Overwrite

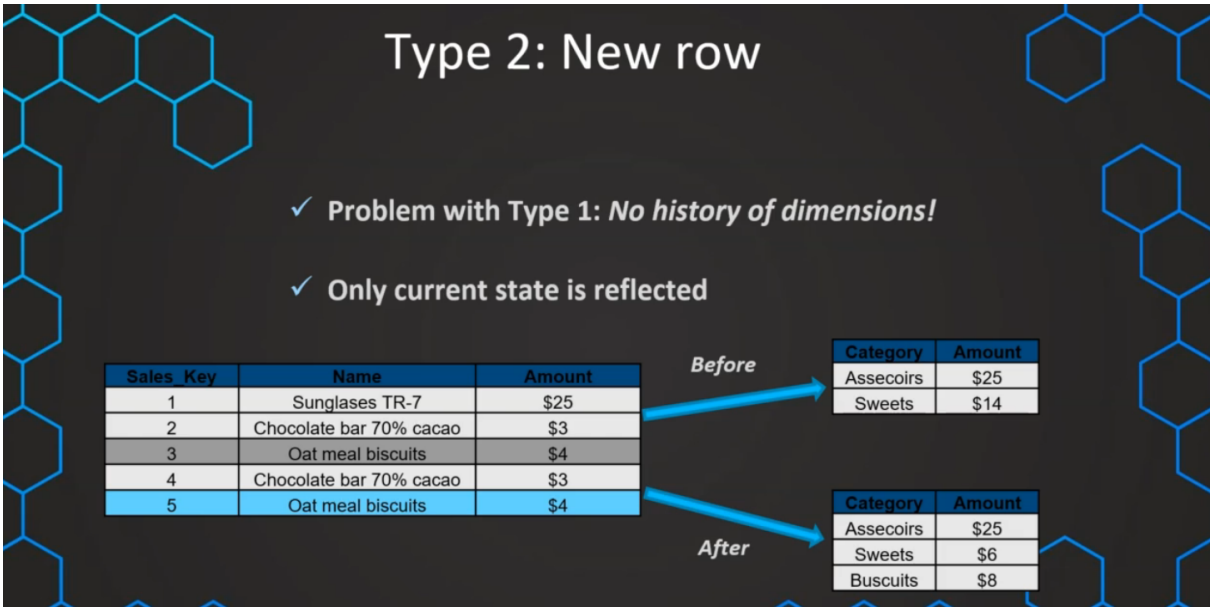


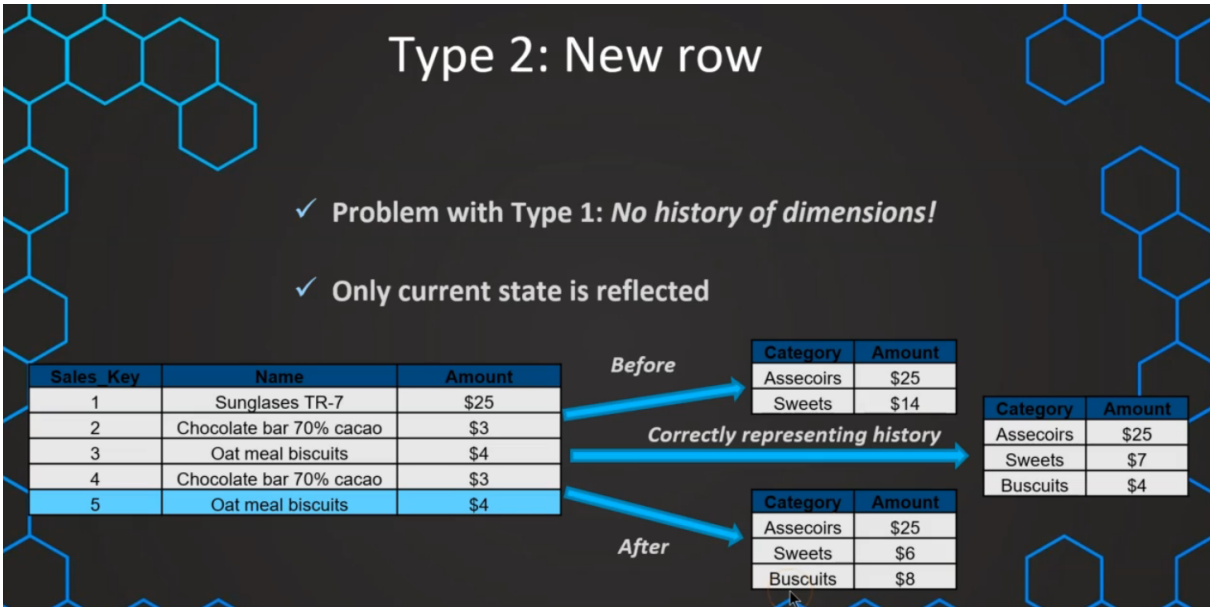


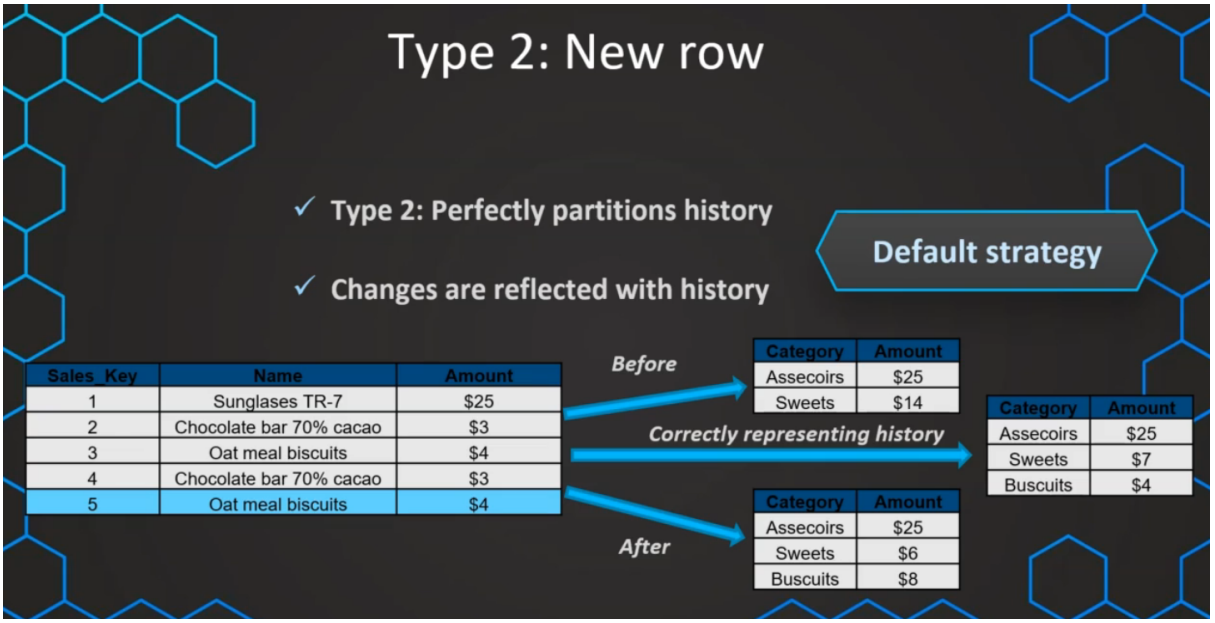
- Suppose we have a group by on Category then now we might see changes as 2 products from Sweets categroy is changed to just 1 product which might affect some aggregated values, also if we have hardcoded in any SQL case statements then we will have to re validate and fix if anything is broken

Type 2: Additional row

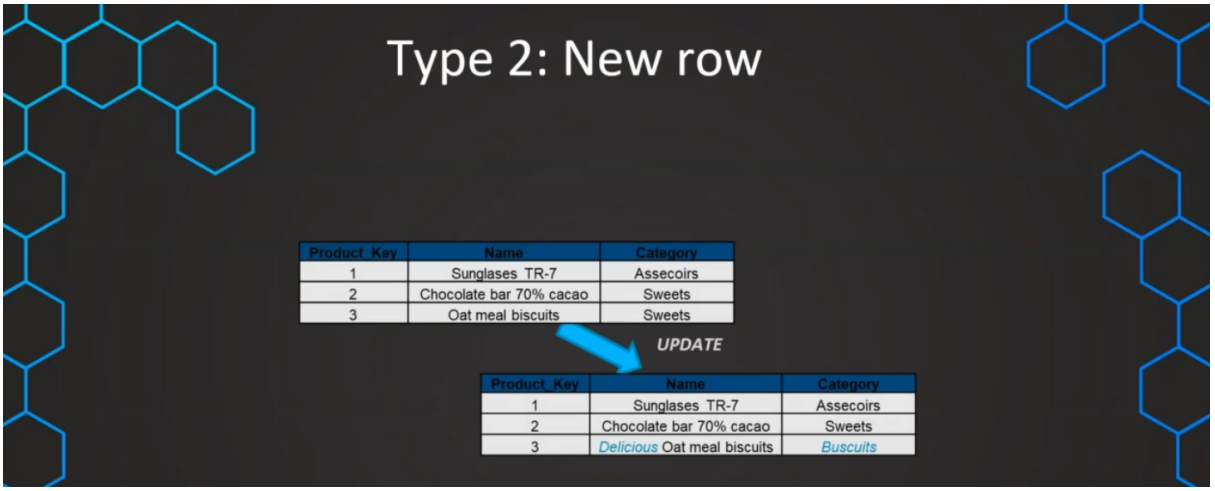


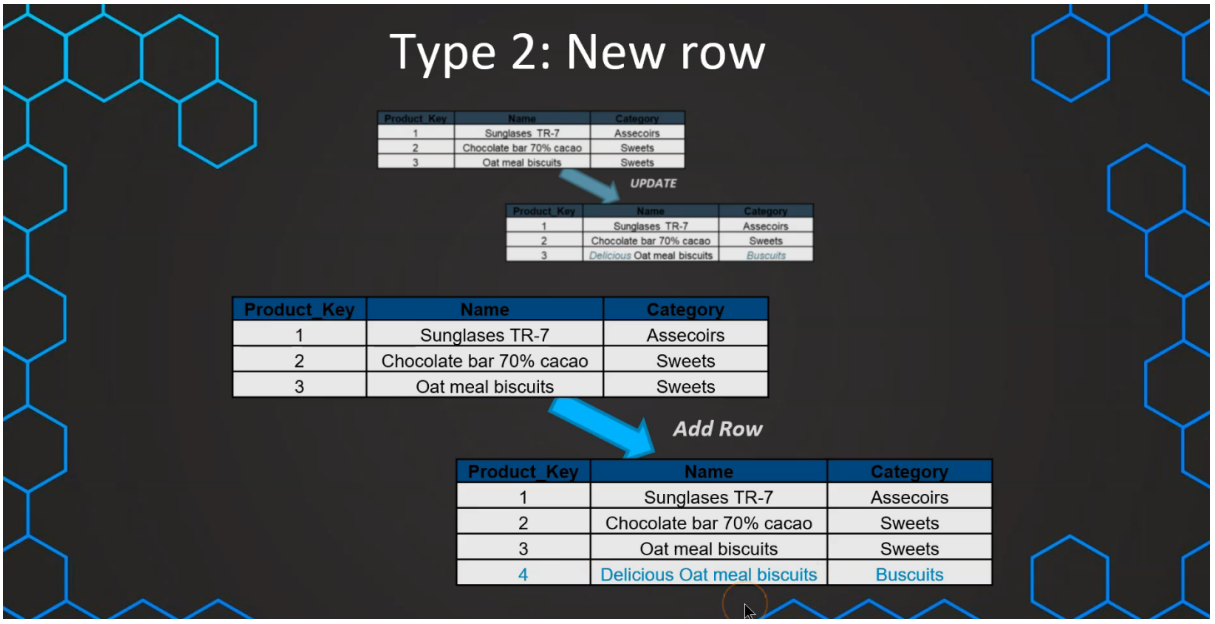


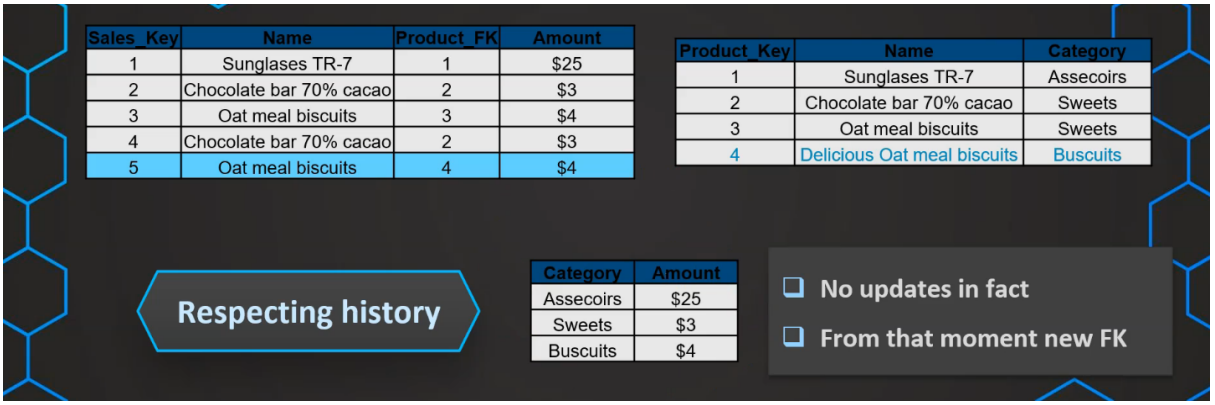


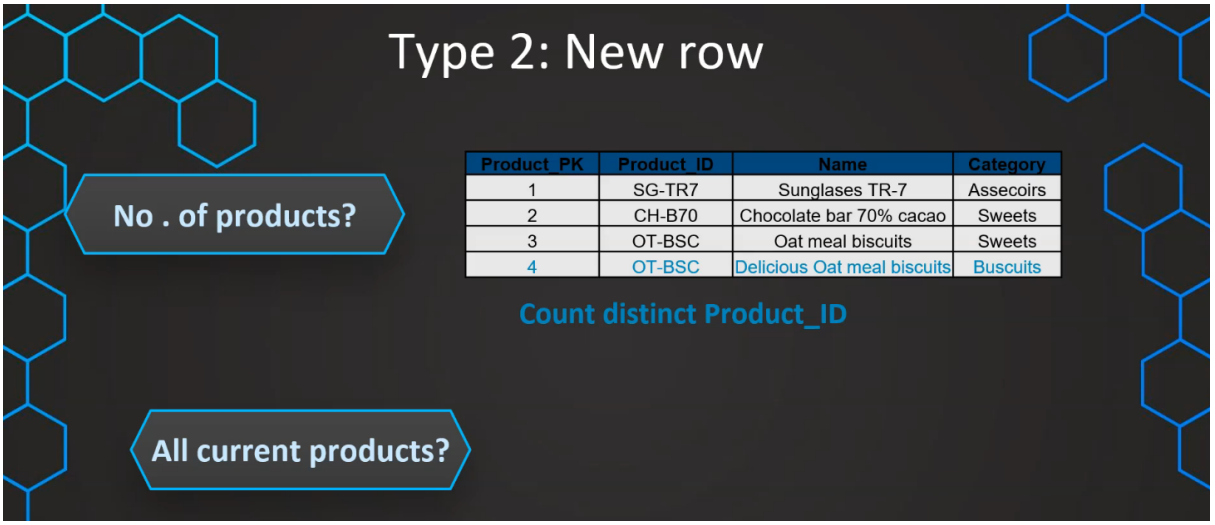


- History is respected i.e. till the dimension is changed, it is calculated to old category , once the dimension is updated with new row, remaining rows are chosing the Biscuits category


Type 2: New row
Product_Key
Name
Category
1
Sunglases TR-7
Assecoirs
2
Chocolate bar 70% cacao
Sweets
3
Oat meal biscuits
Sweets
UPDATE
Product_Key
Name
Category
1
Sunglases TR-7
Assecoirs
2
Chocolate bar 70% cacao
Sweets
3
Delicious Oat meal biscuits
Buscuits


Type 2: New row
Product_Key
Name
Category
1
Sunglases TR-7
Assecoirs
2
Chocolate bar 70% cacao
Sweets
3
Oat meal biscuits
Sweets
UPDATE
Product_Key
Name
Category
1
Sunglases TR-7
Assecoirs
2
Chocolate bar 70% cacao
Sweets
3
Delicious Oat meal biscuits
Buscuits
Product_Key
Name
Category
1
Sunglases TR-7
Assecoirs
2
Chocolate bar 70% cacao
Sweets
3
Oat meal biscuits
Sweets
Add Row
Product_Key
Name
Category
1
Sunglases TR-7
Assecoirs
2
Chocolate bar 70% cacao
Sweets
3
Oat meal biscuits
Sweets
4
Delicious Oat meal biscuits
Buscuits

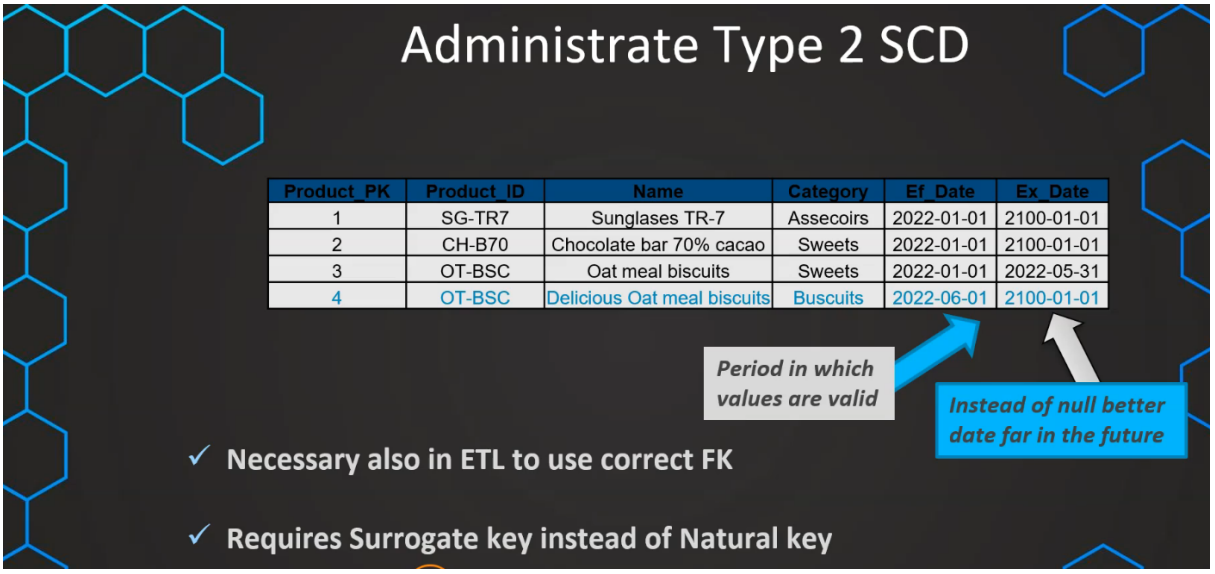

Sales_Key
Name
Product_FK
Amount
1
Sunglases TR-7
1
$25
2
Chocolate bar 70% cacao
2
$3
3
Oat meal biscuits
3
$4
4
Chocolate bar 70% cacao
2
$3
5
Oat meal biscuits
4
$4
Product_Key
Name
Category
1
Sunglases TR-7
Assecoirs
2
Chocolate bar 70% cacao
Sweets
3
Oat meal biscuits
Sweets
4
Delicious Oat meal biscuits
Buscuits
Respecting history
Category
Amount
Assecoirs
$25
Sweets
$3
Buscuits
$4
No updates in fact
From that moment new FK

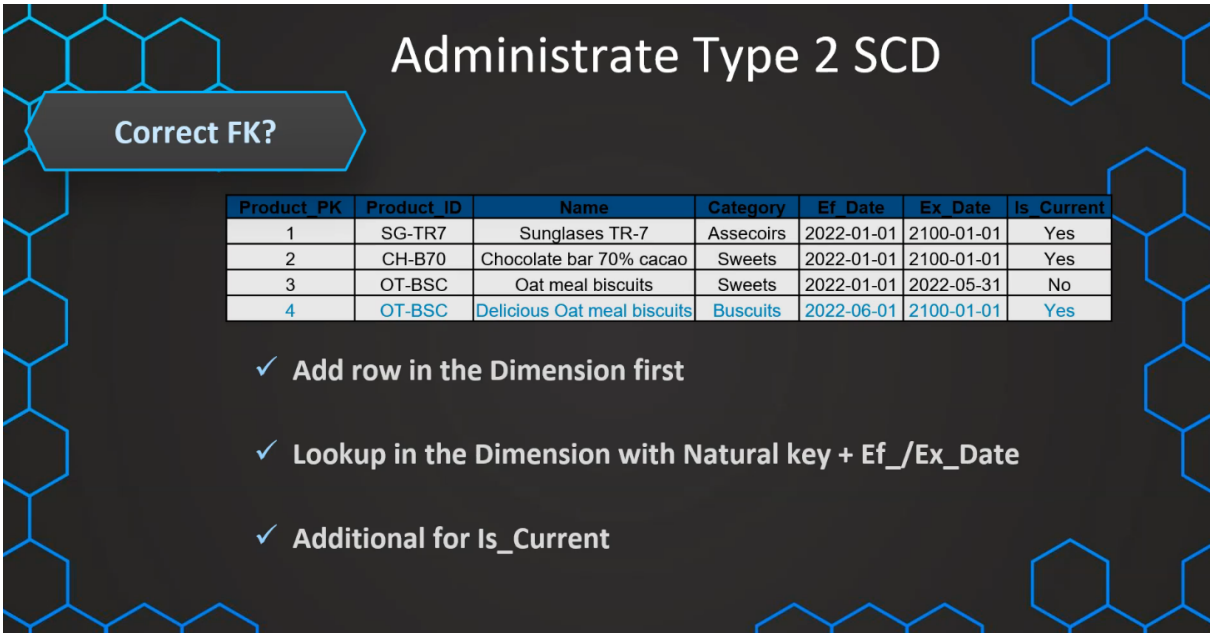


- With the new row added if we want to get number of products, then we can take the distinct product ID which won't affect the new row added which is correct as this is the latest version added

Administrating Type 2 dimensions

- we've seen that this type 2 slowly changing dimension is super powerful because with that we can keep track of the history in our dimensions.
- But unfortunately we've also seen that it's not perfect yet and that we should do some additional things in order to really effectively use this slowly changing dimension.
- Because with current approach we could not identify the current list of updated products as product pk 3 and 4 have same product ID which do not tell which is the latest
- So to overcome this we can make use of effective date and expiration date
- For expiration use a large year like 100-200 years far, and do not use null because if we use nulls then we can't use BETWEEN function effectively
- Effective and Expiration date column is very important for the ETL process to pick the right product



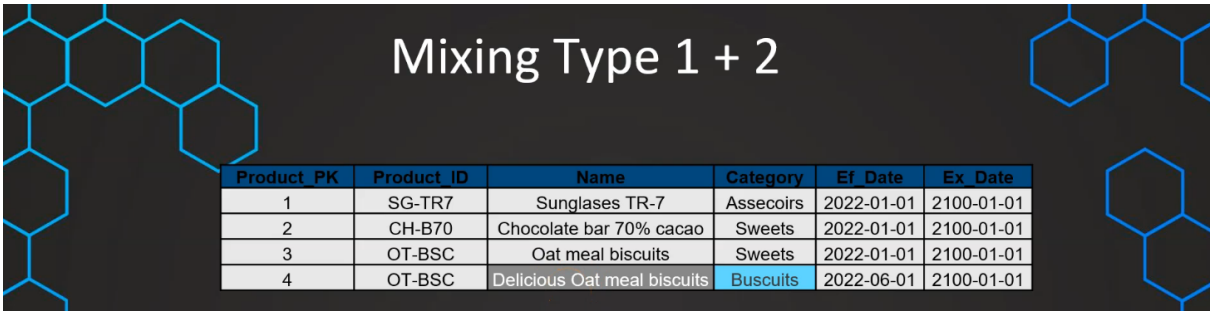


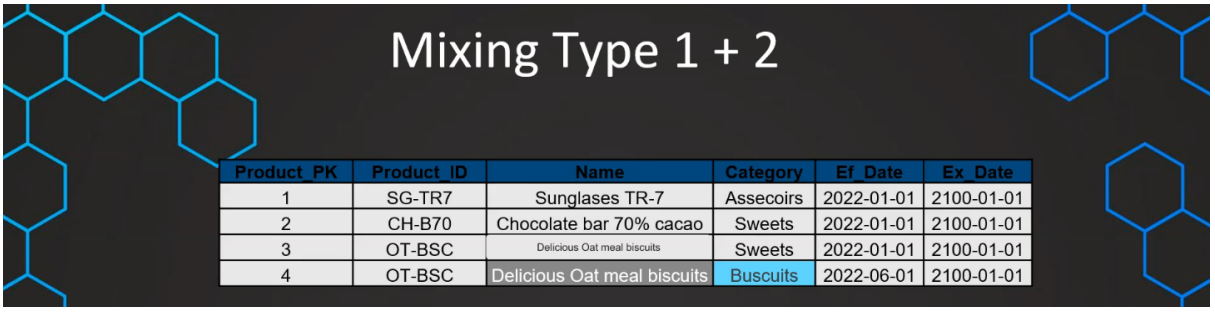
- We can use a natural key along with the help of effective/expiration date column like for a given product PK check if current date time is between effective/ expiration date then only use it
- Other way is we can have a flag column called is_current which says whether a particular row is valid or not

Mixing Type 1 and Type 2

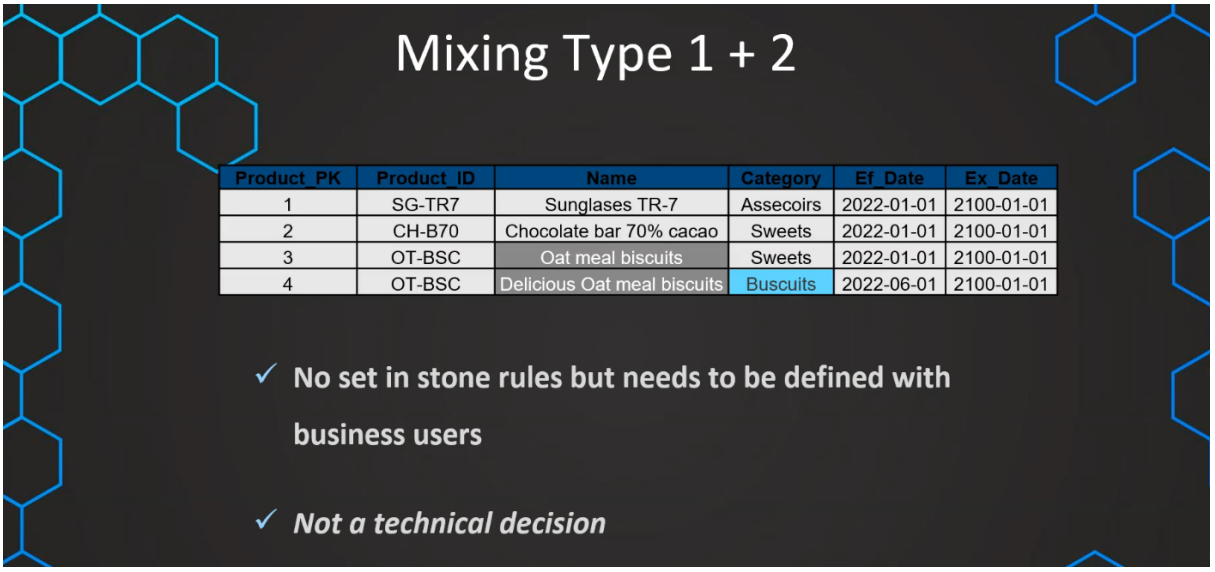
- Now the question might be if we have type 1 and type 2 attributes, do we need to decide for a given dimension table to either use type 1 or type 2? And the answer luckily is NO

- we do not need to decide this for the entire dimension table, but it is depending on the attributes.
- So on the single attributes and some of them can be indeed type 1. So for example, for our product name, it is not really important to have all of the history tract of the product name changes. It is completely fine in our analysis to just override the values and use for all of the history just the same value.





- And this is now not always the case, for example, with more major changes, such as the category. With that, we might want to also track this change and also keep the history.
- So in that case, for the category, we could use type 2. So some of the attributes indeed can be type 1 and some of them can be type 2.
- But what we should keep in mind is that those are not set in stone rules. And also it is not a technical decisions that we should make as data modelers or as designers of the data warehouse.

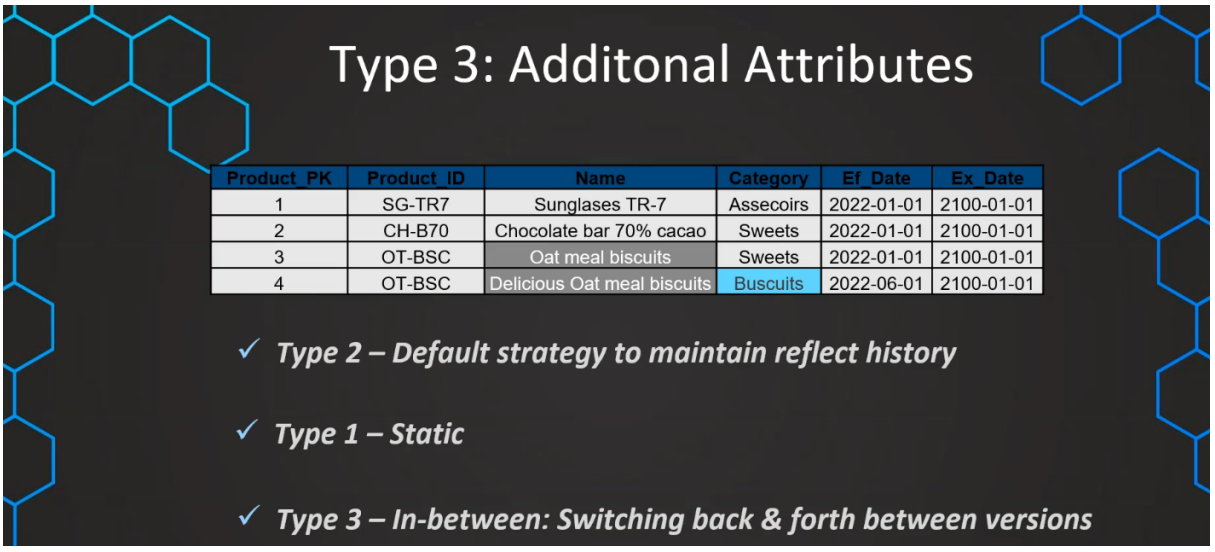


- But this is mainly up to the business users because they need to define, and of course, we can support them and discuss the various options with them. But in the end, they need to know whether they want to keep track of the history or whether it is okay, for example, for the product name, if it is just overwritten and therefore we use type 1.
- So this is important that we talk to the business users and we define those roles together with those business users. And of course the business user in the end, they need to use this data.
- And therefore, of course the last word is of course with those business users.
- In some situations, both of these type 1 and type 2 are not the right fit and we have a better solution for that, which is now type number 3.

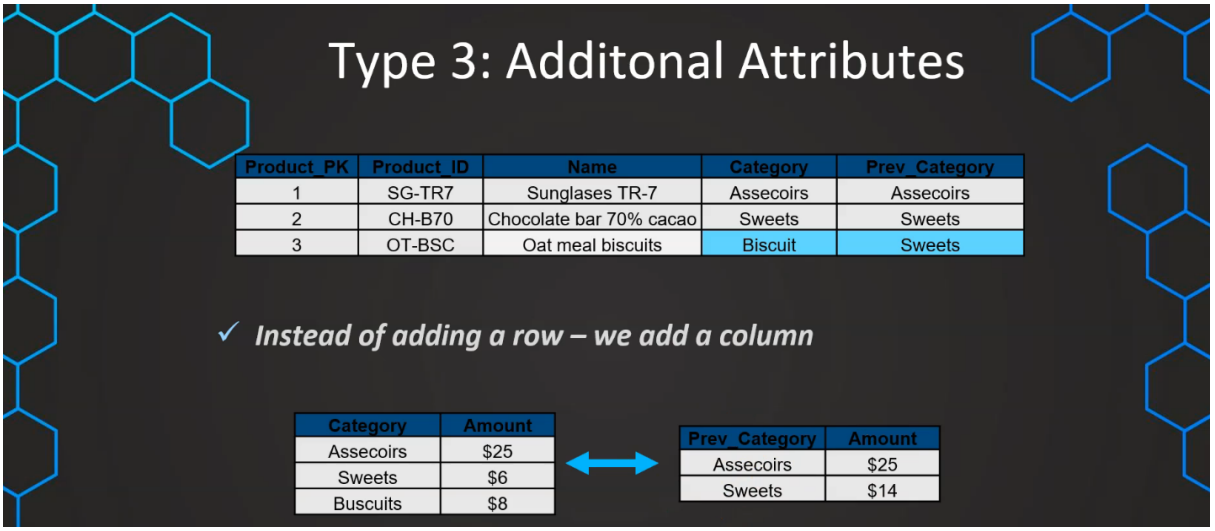
Type 3: Additional attributes

- Now apart from type two and type one there's also this type three, slowly changing dimension.
- But now how is this different from type one and type two?
- So far we've learned about type two which is if we have many changes and we want to have the entire history of our changes in the dimension reflected, this type two strategy and this is the default strategy if we want to have the entire history maintained.

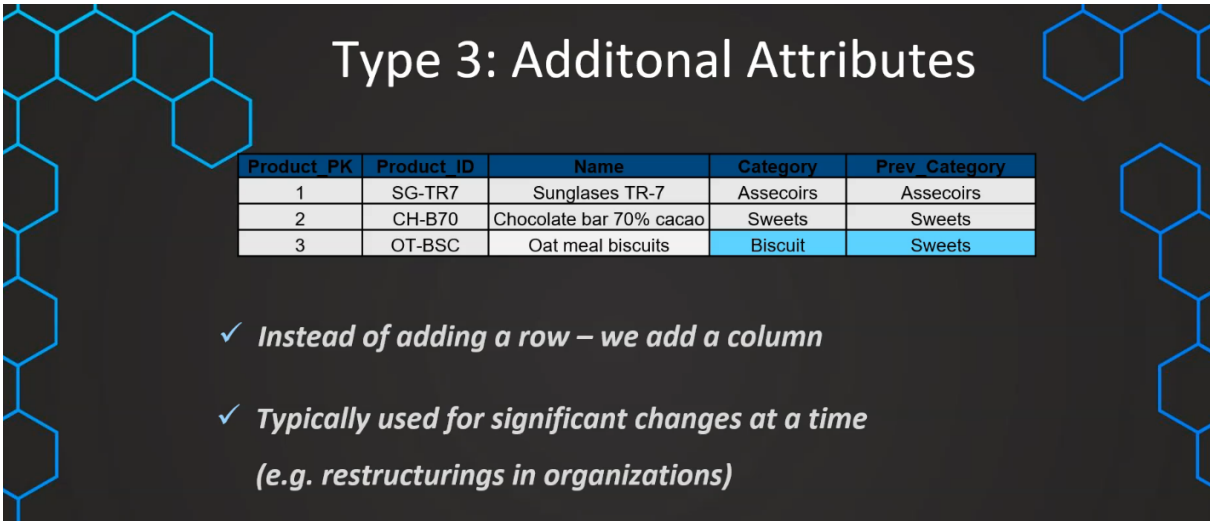
- And then on the other side we've also learned about type one which is completely static because it's just reflecting the current state of our dimension.
- And now with type three, we have something that is somewhat in between because now we have different versions. So usually we have two different versions and we can switch back and forth between them.
- So for example we can have the static current state, and then the second state is again the static previous state and we can switch back and forth between current and historic view.



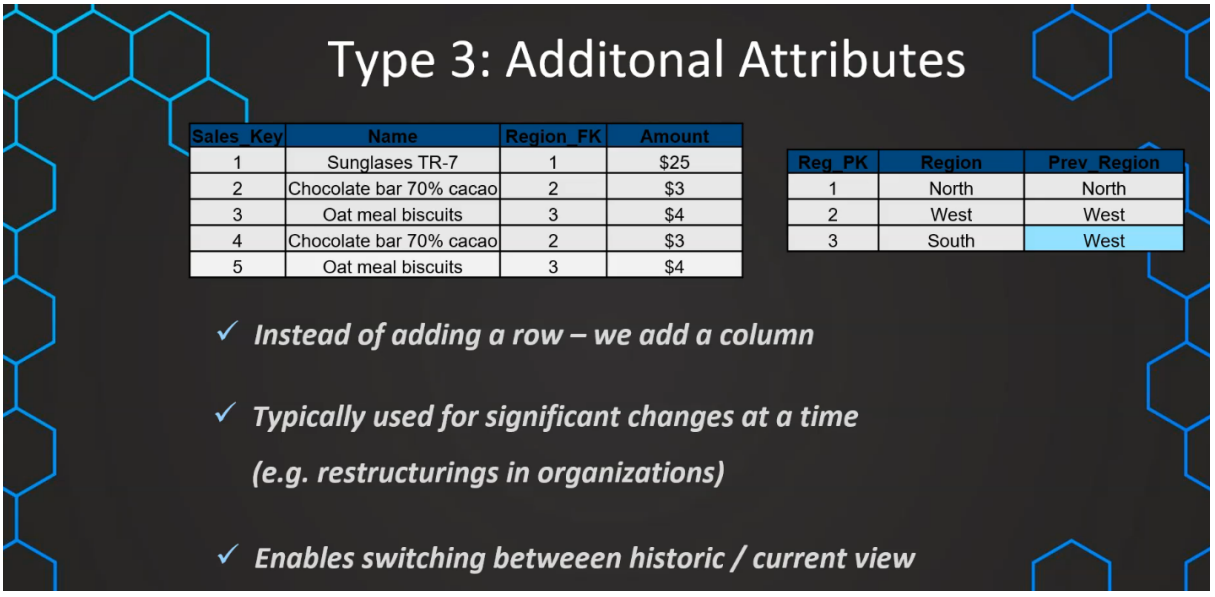
- So this is now something that is somewhat in between but now how is this implemented?
- And this is also now different from the previous implementations because now instead of adding an additional row we want to add an additional column. And this additional column is just now containing the previous value. So this is basically now the historical value.
- And in the normal categorical value we just have the current state. So with those two columns we can now reflect the two different states.



- And the great thing is that now we cannot only use that category in our analysis. So this is the current state but we can switch also to the previous state and aggregate the data with this previous state.
- And this is now something that is used usually if we have some significant major changes that are usually planned and are happening all at one time.
- A typical example is the restructuring in an organization.

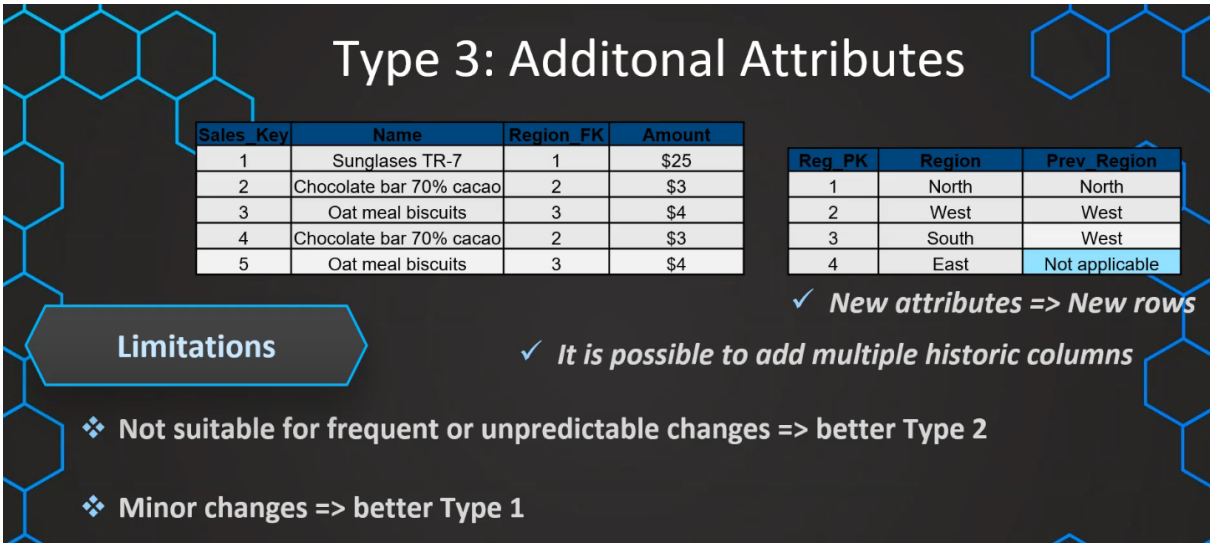


- So for example, we've decided in the company that all of the categories are restructured or maybe also the hierarchies in our organization have changed the regions are now different and then we can implement the strategy.
- And this is now great because with that we can now switch back and forth between the previous historic state and the current state.



- So imagine you are working as a manager and you want to analyze the sales by region but you would also maybe after the restructuring would like to find out how would your numbers be if the restructuring would not have happened.
- So for example, previously there was only north and west and now there's also south introduced. And in that case you would also like to be able to switch back also to this previous state.
- And then you can just use this previous region column to group your data with this previous state. And now both of these states are reflected and you can easily even switch back and forth between them.
- So in that situation when there's some big restructuring happening all at one time, then this is an appropriate strategy and not also that we are very flexible in just also adding an additional row if there are some new attributes and there is not a something that can be associated with that in the past.
- So for example, a new product line and we did not have something associated to that in the past. In that case, we can also just easily add an additional row and have now this new attribute added.
- And also it is possible to not only add one additional row but if we have a change, again, some again some restructuring for example then we can also add a second or third additional column to just also reflect then more than just those two different states.
- Of course, we don't want to bring this into a very extreme so that we have now hundreds of columns.

- This is not really the idea of that, but in general it is possible to just have more than just two states if it should be necessary.
- But now again, there are some limitations and this type three is not always even actually in most cases not really the best fit because it's really just good for one specific situation.
- And this is usually when we have a big change happening that is very predictable.



- So for example, one restructuring and this is happening very predictable. One moment, all of the values are changing.
- If we have more frequent changes and they are happening very unpredictably then we should better just go with type two.
- And on the other side, if the changes are really just minor changes that nobody really cares about, for example a change in a product name, then we can also just better use type one because it's the easiest.
- And this is then in such situations the best one.
- So therefore, even though this is an in-between solution or in-between strategy in reality it is good for a specific situation.
- And that is when we have big changes that are predictable and the users want to be able to switch back and forth between different states.
- And this is the situation when we want to use this type three, slowly changing dimension.

Quiz

Good job!
A perfect partition of the history we can achieve using SCD Type 2 by adding a new row for every update/insert of a new dimension value.

Question 1:

What type of slowly changing dimensions should we use if the changes can be more frequent and occur not really predictable?

- ○ Type 0
- ○ Type 1
- ◉ **Type 2**
- ○ Type 3

Question 2:

If there is are columns like "Effective date" and "Expiry date" in a given dimension. What type of slowly changing dimension are we dealing with?

- ○ Type 0
- ○ Type 1
- ◉ **Type 2**
- ○ Type 3

Question 3:

Given that it is not important to maintain any history of changes of a given dimension but we prefer to use a simple method, what type of SCD can you implement?

- (●) **Type 1**
- ( ) **Type 2**
- ( ) **Type 3**