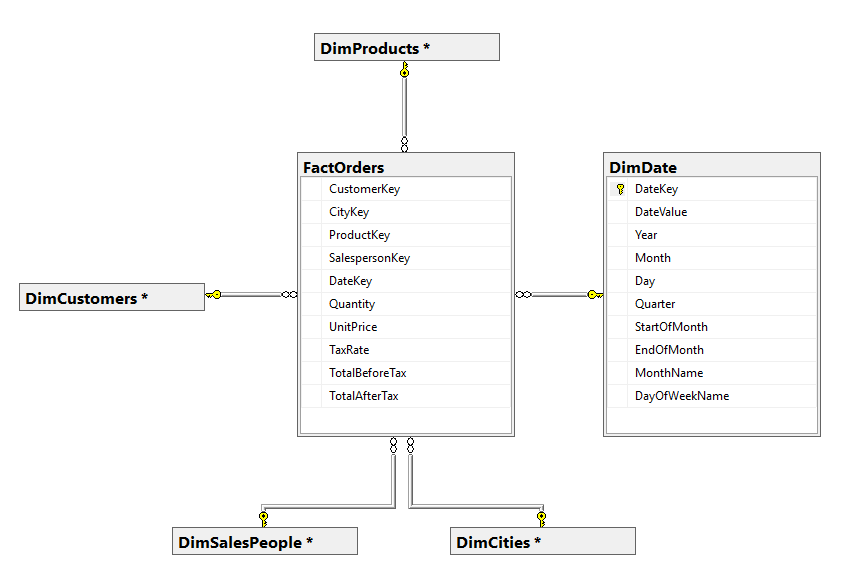
Slowly Changing Dimensions

# Dimension Review

## Example Model

We have completed the basics of our model and fleshed out a few measures in our fact table. We have also completed our date dimension. Before we continue, let’s review our model:

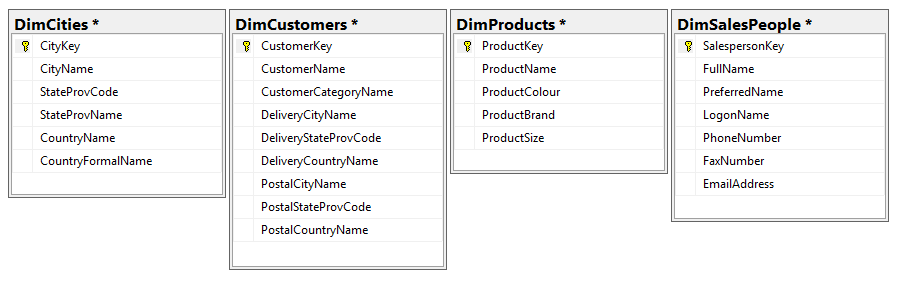


To recap:

* In a **star schema** dimensional model, all foreign key relationships have a **fact table as the child** and a **dimension table as the parent**.
* We added keys in the fact table for each of our dimensions. The combination of those keys define the **granularity** for the fact table.
* We did not add a surrogate key or a primary key to our fact table. It is common to implement **fact tables as a heap**.
* We created a date dimension and gave it an integer **smart key** as the primary key. We created this based on the date value in YYYYMMDD format.
* We created several columns in the date dimension that are easily calculable based on the date value. This is normal in a data mart, where we want to prioritize performance over eliminating redundancy.

In our previous classes, we’ve also identified a few columns that are likely a good fit for our dimension tables.

We can see those columns added to our tables below:



When we looked at the date dimension, we noted that it would not be meaningful for any of the values to ever change. What about these tables? Can the properties of a record change, but still be the same record?

Is it reasonable, for example, for a salesperson to get a new phone number but still be the same salesperson?

# Slowly Changing Dimensions

## Basics

As we know, fact tables usually represent a point in time, such as an event or a transaction. We have also learned that fact tables are defined by their dimensions; the combination of all of the dimension keys is a unique key for the fact table. This means that the dimension keys cannot change – if any of the keys change, it’s a different fact record.

As a result, there are usually no columns in a fact table that are meaningful to change. Once we create a fact record, we do not alter it.

In contrast, we have learned that dimensions are more persistent. One dimension record may be associated with many fact records over time. The properties of the dimension help us better define the facts that they are associated with.

A dimension record is uniquely defined by a natural key, which we usually refer to as a **business key** in data warehousing. Any of the other properties might change over time. We refer to this as a **slowly changing dimension** (SCD).

A **slowly changing dimension** is a dimension where non-key attributes may change over time.

We will learn about the three most common types of SCD.

## Type 0 SCD

In a Type 0 SCD, records may never change. We may only add new records. Any attempt to change an existing record in our ETL should result in an error.

We have seen this type of SCD already with our date dimension.

In our date dimension, our business key was DateValue. Every other column in our table was based on this column, making it impossible for our non-key attributes to change. If our process ever tried to change any of those attributes, it would be an error.

A **Type 0 SCD** is a dimension where non-key attributes may never change.

## Type 1 SCD

In a Type 1 SCD, non-key attributes are changed by overwriting the original value. This means that all associated fact records will be associated with the new value, regardless of what the value was at the time the fact record was created.

For example, in budget system, we might have a department dimension that has a department code as the business key and department name as a non-key attribute. We don’t care about tracking all the changes in the department name, we just need to know the name as it currently exists.

If we had this table associated with an expenses fact table, it may look like the following:

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| **DimDepartment** | | |  | **FactExpenses** | | |
| **DepKey** | **Code** | **Name** |  | **DepKey** | **DateKey** | **Amount** |
| 1 | HR | Human Resources |  | 1 | 20171201 | 103.17 |
| 2 | ENG | Engineering |  | 1 | 20171202 | 0.00 |
|  |  |  |  | 2 | 20171201 | 58.98 |
|  |  |  |  | 2 | 20171202 | 16.83 |

|  |  |  |  |
| --- | --- | --- | --- |
| **Report of Expenses by Department by Date** | | | |
| **Code** | **Name** | **DateKey** | **Amount** |
| HR | Human Resources | 20171201 | 103.17 |
| HR | Human Resources | 20171202 | 0.00 |
| ENG | Engineering | 20171201 | 58.98 |
| ENG | Engineering | 20171202 | 16.83 |

In the above example, we have two records in our FactExpenses table that are associated with the engineering department.

We created a report by using simple join that would look something like the following:

SELECT d.code,

d.name,

f.datekey,

f.amount

FROM dbo.DimDepartment d

JOIN dbo.FactExpenses f

ON d.DepKey = f.DepKey;

When we join these tables to create a report, we see “Engineering” associated with the appropriate expenses for December 1st and December 2nd.

If on December 3rd, the name of the department changed, it would look like this:

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| **DimDepartment** | | |  | **FactExpenses** | | |
| **DepKey** | **Code** | **Name** |  | **DepKey** | **DateKey** | **Amount** |
| 1 | HR | Human Resources |  | 1 | 20171201 | 103.17 |
| 2 | ENG | Software Engineering |  | 1 | 20171202 | 0.00 |
|  |  |  |  | 1 | 20171203 | 28.45 |
|  |  |  |  | 2 | 20171201 | 58.98 |
|  |  |  |  | 2 | 20171202 | 16.83 |
|  |  |  |  | 2 | 20171203 | 108.75 |

|  |  |  |  |
| --- | --- | --- | --- |
| **Report of Expenses by Department by Date** | | | |
| **Code** | **Name** | **DateKey** | **Amount** |
| HR | Human Resources | 20171201 | 103.17 |
| HR | Human Resources | 20171202 | 0.00 |
| HR | Human Resources | 20171203 | 28.45 |
| ENG | Software Engineering | 20171201 | 58.98 |
| ENG | Software Engineering | 20171202 | 16.83 |
| ENG | Software Engineering | 20171203 | 108.75 |

Since DimDepartment is a Type 1 SCD, we just overwrite “Engineering” with the new value “Software Engineering”.

When we create the same report on January 3rd, all of our records associated with the ENG department now show “Software Engineering” even though the same report showed a different name for this department the previous day.

A **Type 1 SCD** changes non-key attributes by overwriting the existing value. No history of changes is retained.

## Type 2 SCD

In a Type 2 SCD, non-key attributes are changed by creating a new record. This allows us to preserve a history of the changes. It also allows us to preserve the relationship between facts and the dimension values they were associated with when the fact records were created.

For example, in an HR system, we might an employee dimension that has employee number as the business key. We might track name and marital status as non-key attributes. These non-key attributes can change, but it is important for our reporting that we don’t change any existing associations.

We have this dimension associated with a benefit claims fact, which we have reported on using a simple join.

|  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- |
| **DimEmployee** | | | |  | **FactClaims** | | |
| **EmpKey** | **EmpNo** | **Name** | **Status** |  | **EmpKey** | **DateKey** | **Amount** |
| 1 | RB00451 | Ray Bradbury | M |  | 1 | 20171201 | 12.01 |
| 2 | IA01101 | Isaak Ozimov | S |  | 2 | 20171201 | 6.80 |
| 3 | GO01984 | George Orwell | S |  | 1 | 20171202 | 91.55 |
|  |  |  |  |  | 2 | 20171202 | 58.00 |
|  |  |  |  |  | 3 | 20171202 | 12.98 |

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **Report of Claims by Employee** | | | | |
| **EmpNo** | **Name** | **Status** | **DateKey** | **Amount** |
| RB0451 | Ray Bradbury | M | 20171201 | 12.01 |
| IA01101 | Isaak Ozimov | S | 20171201 | 6.80 |
| RB0451 | Ray Bradbury | M | 20171202 | 91.55 |
| IA01101 | Isaak Ozimov | S | 20171202 | 58.00 |
| GO01984 | George Orwell | S | 20171202 | 12.98 |

If on December 3rd, employee IA01101 change their name and marital status, we would want to preserve the associations that already existed on December 1st and 2nd. Instead of updating the existing record for that employee, we would create a new one using the same business key, resulting in the following:

|  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- |
| **DimEmployee** | | | |  | **FactClaims** | | |
| **EmpKey** | **EmpNo** | **Name** | **Status** |  | **EmpKey** | **DateKey** | **Amount** |
| 1 | RB00451 | Ray Bradbury | M |  | 1 | 20171201 | 12.01 |
| 2 | IA01101 | Isaak Ozimov | S |  | 2 | 20171201 | 6.80 |
| 3 | GO01984 | George Orwell | S |  | 1 | 20171202 | 91.55 |
| 4 | IA01101 | Isaac Asimov | M |  | 2 | 20171202 | 58.00 |
|  |  |  |  |  | 3 | 20171202 | 12.98 |
|  |  |  |  |  | 3 | 20171203 | 280.15 |
|  |  |  |  |  | 4 | 20171203 | 60.01 |

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **Report of Claims by Employee** | | | | |
| **EmpNo** | **Name** | **Status** | **DateKey** | **Amount** |
| RB0451 | Ray Bradbury | M | 20171201 | 12.01 |
| IA01101 | Isaak Ozimov | S | 20171201 | 6.80 |
| RB0451 | Ray Bradbury | M | 20171202 | 91.55 |
| IA01101 | Isaak Ozimov | S | 20171202 | 58.00 |
| GO01984 | George Orwell | S | 20171202 | 12.98 |
| GO01984 | George Orwell | S | 20171203 | 280.15 |
| IA01101 | Isaac Asimov | M | 20171203 | 60.01 |

Notice that we have a new record with a new surrogate key, but the same business key. The fact table references the dimension’s surrogate key, so it is always associated with the version that existed when the fact record was written.

A **Type 2 SCD** changes non-key attributes by creating a new record. This way a historical record is retained and fact tables remain associated with the dimension record that existed when the fact record was created.

## Type 2 Effective Dates

Our example above is incomplete. If we were trying to query it, how would we know which version of IA01101 is the current one? When did it become the active version? When was the previous version active?

To resolve this issue, we need to add effective dates to the table. There are many naming conventions for these dates, but we will be using StartDate and EndDate.

Let’s review our example above again, but this time we will add our effective dates and focus only on one employee.

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| **EmpKey** | **EmpNo** | **Name** | **Status** | **StartDate** | **EndDate** |
| 2 | IA01101 | Isaak Ozimov | S | 2017-12-01 | NULL |

We first added IA01101 to our table on December 1st. Since there was no previous record of IA01101, we inserted a new record. We record 2017-12-01 as the first day this record was active. We do not yet know when this record will expire, so we record the end date as NULL.

In our example above, we changed the name and marital status for this record on December 3rd.

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| **EmpKey** | **EmpNo** | **Name** | **Status** | **StartDate** | **EndDate** |
| 2 | IA01101 | Isaak Ozimov | S | 2017-12-01 | 2017-12-02 |
| 4 | IA01101 | Isaac Asimov | M | 2017-12-03 | NULL |

To add a new current record for IA01101, we have to expire the existing record. We do that by recording December 2nd, the day before the new record became active, as the last day the old record was active. We then insert our new record, with December 3rd as the start date.

Let’s change this record one more time. On December 14th, this employee’s status is going to change back to single.

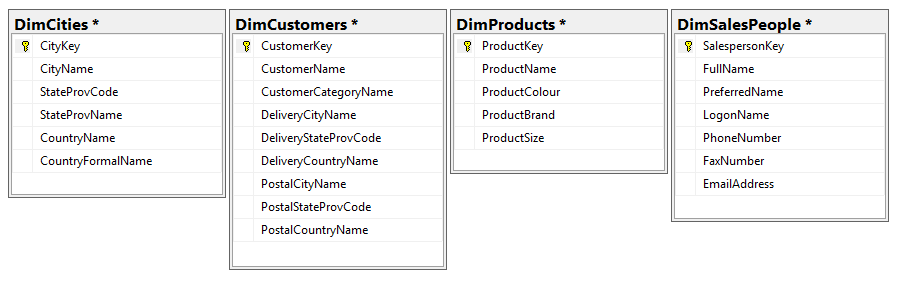
|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| **EmpKey** | **EmpNo** | **Name** | **Status** | **StartDate** | **EndDate** |
| 2 | IA01101 | Isaak Ozimov | S | 2017-12-01 | 2017-12-02 |
| 4 | IA01101 | Isaac Asimov | M | 2017-12-03 | 2017-12-13 |
| 5 | IA01101 | Isaac Asimov | S | 2017-12-14 | NULL |

Like our last change, we expire the previous record by adding the end date, then add a new record.

## Choosing Types

Let’s review our four incomplete dimensions to see which type of SCD each should be. To do so, we should identify the business key for each table so that we can identify non-key attributes. As a reminder, we can pick an SCD type using the following criteria:

Type 0 – Should never change. Changes are an error.  
Type 1 – Can change. No history needed. Changes do not alter the meaning of associated facts.  
Type 2 – Can change. History required. Changes may alter the meaning of associated facts.



**DimCities**

This table is based on Application.Cities, Application.StateProvinces, and Application.Countries. Examining these tables, we can see that CityName is not unique, which makes sense. However, given the structure of our table, it is reasonable to treat a City-StateProvince-Country combination as a business key. So our business key is **CityName, StateProvName, CountryName**.

The remaining columns could theoretically change. It does not seem likely that such changes would alter the meaning of associated facts. We can treat this table as a **Type 1 SCD**. The table is complete as-is.

**DimCustomers**

This is based primarily on Sales.Customers which has **CustomerName** as a unique key, so this would work well as our business key.

The remaining columns could change and would definitely alter the interpretation of associated facts. For example, if a customer moved from Ontario to Quebec, we would not want all of their historical purchases to be switched to the new address. We can treat this table as a **Type 2 SCD.** We will need to add StartDate and EndDate to this table.

**DimProducts**

This is based primarily on Warehouse.StockItems, which also has a unique key, StockItemName. In our table this unique key will be renamed **ProductName** and can be used as our business key.

The remaining columns can change and would alter the interpretation of associated facts. We should treat this as a **Type 2 SCD.** We will need to add StartDate and EndDate to this table.

**DimSalesPeople**

DimSalesPeople is based on Application.People, which has no unique key. However, examining the table we can see that LogonName is unique and always populated for salespeople. In practice, we would have to check with our business analyst to see if this is reasonable, but for our purposes we will be using **LogonName** as our business key.

The remaining columns could change, but it is debatable whether these changes would impact associated facts. For our purposes we will treat it as a **Type 1 SCD**, so the table needs no extra columns.

## Completed Model

Adding our effective date columns gives us the following DDL for our final four tables.

CREATE TABLE dbo.DimCities(

CityKey INT NOT NULL,

CityName NVARCHAR(50) NULL,

StateProvCode NVARCHAR(5) NULL,

StateProvName NVARCHAR(50) NULL,

CountryName NVARCHAR(60) NULL,

CountryFormalName NVARCHAR(60) NULL,

CONSTRAINT PK\_DimCities PRIMARY KEY CLUSTERED ( CityKey )

);

CREATE TABLE dbo.DimCustomers(

CustomerKey INT NOT NULL,

CustomerName NVARCHAR(100) NULL,

CustomerCategoryName NVARCHAR(50) NULL,

DeliveryCityName NVARCHAR(50) NULL,

DeliveryStateProvCode NVARCHAR(5) NULL,

DeliveryCountryName NVARCHAR(50) NULL,

PostalCityName NVARCHAR(50) NULL,

PostalStateProvCode NVARCHAR(5) NULL,

PostalCountryName NVARCHAR(50) NULL,

StartDate DATE NOT NULL,

EndDate DATE NULL,

CONSTRAINT PK\_DimCustomers PRIMARY KEY CLUSTERED ( CustomerKey )

);

CREATE TABLE dbo.DimProducts(

ProductKey INT NOT NULL,

ProductName NVARCHAR(100) NULL,

ProductColour NVARCHAR(20) NULL,

ProductBrand NVARCHAR(50) NULL,

ProductSize NVARCHAR(20) NULL,

StartDate DATE NOT NULL,

EndDate DATE NULL,

CONSTRAINT PK\_DimProducts PRIMARY KEY CLUSTERED ( ProductKey )

);

CREATE TABLE dbo.DimSalesPeople(

SalespersonKey INT NOT NULL,

FullName NVARCHAR(50) NULL,

PreferredName NVARCHAR(50) NULL,

LogonName NVARCHAR(50) NULL,

PhoneNumber NVARCHAR(20) NULL,

FaxNumber NVARCHAR(20) NULL,

EmailAddress NVARCHAR(256) NULL,

CONSTRAINT PK\_DimSalesPeople PRIMARY KEY CLUSTERED (SalespersonKey )

);