Slowly Changing Dimensions made Easy with Durable Keys

Niall Langley

Data Developer / Consultant

11 years experience with SQL Server doing OLTP and data warehousing / BI, now working with Azure data platform
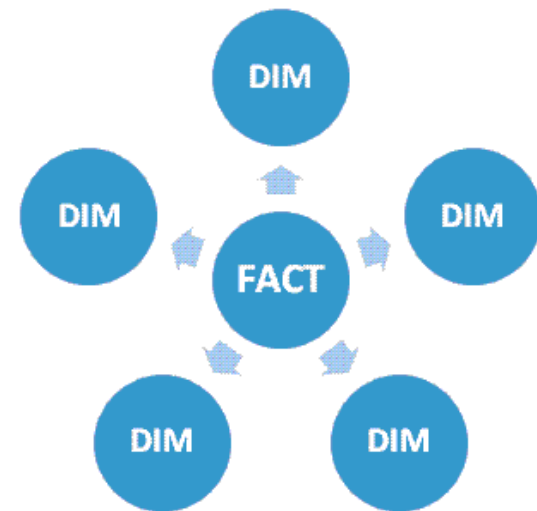
Blog: www.sqlsmarts.com

LinkedIn: uk.linkedin.com/in/niall-langley

Twitter: @NiallLangley

Introduction

- Why is this still relevant with the cloud?
- What problem are we trying to solve?
- What are current solutions?
- How can we solve it better?
- Implementation
- Demo
- Summary

Note: Having a basic understanding of Kimball dimensional modelling is a
pre-requisite for this talk.

BI: As-is **vs** As-posted **vs** As-of

- BI is done using star schemas – facts measure things, and dimensions provide context to those measurements
- Dimensional attributes change over time, such as a customer changing address
- Businesses use those dimensional attributes in three ways
  - As-is – Shows data using current structure
  - As-posted – Shows data using the "historical truth", eg. the structure at the time that data point happened
  - As-of – Shows data using structure at point in time, eg. 1st Jan 2020
- We model As-posted requirements using slowly

  changing dimensions

Examples

- Organisation Hierarchies
  - Reports & dashboards for leadership mostly use the current hierarchy
  - Calculating bonuses for an employee that moved within the hierarchy should use the historical values

- Customer Insights
  - Customers are assigned to buckets based on behavior, eg. 'Infrequent Spender', 'Star Customer'
  - Dashboards focus on current attributes, eg. sales broken out by bucket
  - Analysis focuses on how to convert an 'Infrequent Spender' into a 'Star Customer' and requires historical values
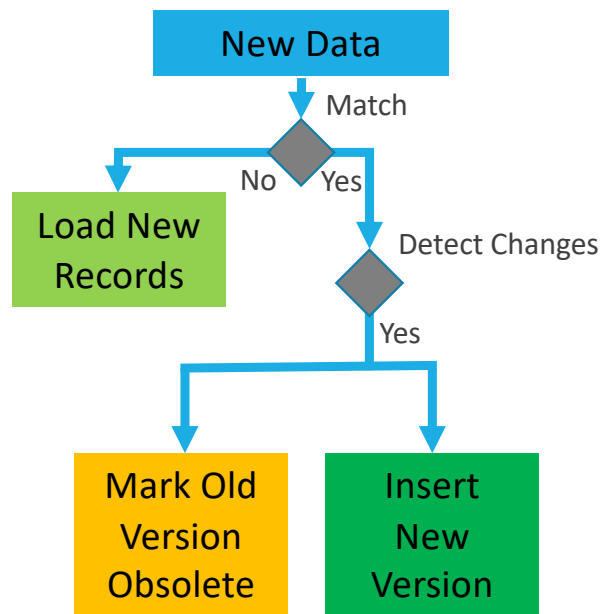
Modelling Slowly Changing Dimensions (SCD)

Type 0	Dimension is never updated. Records are as they were when they were loaded.
Type 1	Dimension is always updated. Records are show the current version of attributes.
Type 2	Dimension stores the history of updates by adding new rows for new versions, with valid from and to dates. Each row representing the history of an entity has a different surrogate key.
Type 3	Dimension stores limited history by using two sets of columns, of for the original or previous value, and one for the current value

Types of keys

- Surrogate keys are assigned in the ETL/ELT. Fact tables link to these.
- Business/Natural keys identify a row from a source system. They are often composite keys.

https://en.wikipedia.org/wiki/Slowly_changing_dimension

SCD Type 2 Example

New Data

Match

No — Yes

Load New Records

Detect Changes

Yes

Mark Old Version Obsolete

Insert New Version

No Change

ID	Name	Description
5	Mint Tea	Refreshing mint tea
9	Breakfast Tea	90% Assam, 10% Ceylon

Mark Old Version Obsolete
Add new row

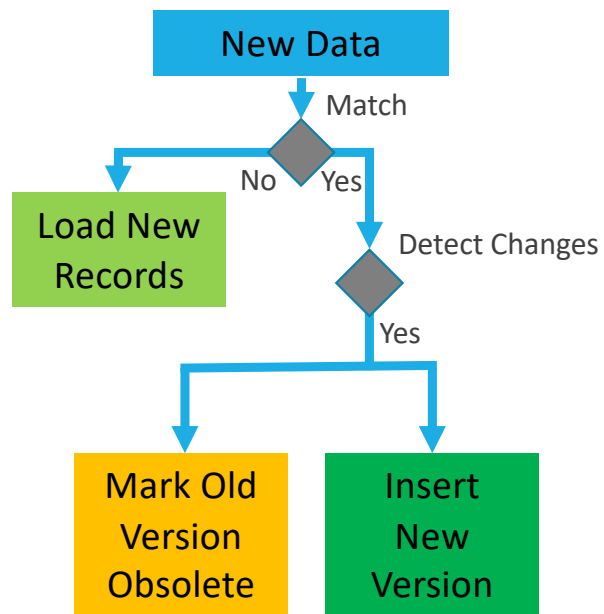| SK	BK	Name	Description	StartDate	EndDate	IsCurrent
1	T5	Mint Tea	Refreshing mint tea	2019-06-22	9999-21-31	1
2	T9	Breakfast Tea	90% Ceylon, 10% Assam	2019-06-22	~~9999-21-31~~ 2020-02-04	~~1~~ 0
3	T9	Breakfast Tea	90% Assam, 10% Ceylon	2020-02-05	9999-21-31	1

What's the problem

- Business users often think they require either as-is **or** as-posted views on the data, not both
- Changing the model later is expensive
- Making all dimensions type 2 SCD is also expensive
  - Development is more complex
  - Viewing the data using the current version requires complex queries
  - Avoiding business key joins is a key principal of the Kimball approach
- Extending SCD Type 2 to add current value columns (Type 6), or updating a column in a fact pointing to the current version adds expensive updates to ETL/ELT

Extending SCD Type 2 with Durable Keys

- They have the properties of a surrogate / primary key, just at the entity level, not the history tracked level
  - Unique - to an entity in the dimension, same across all versions of the entity
  - Consistent – Once set it does not change
  - Minimal – Single integer column
  - Definitive – The value is assigned at row insertion
  - Factless – No smart keys, not a business key, extra surrogate key managed by ETL/ELT process
- Should be easy to retrofit to an existing SCD Type 2
- Simple to understand and implement as a

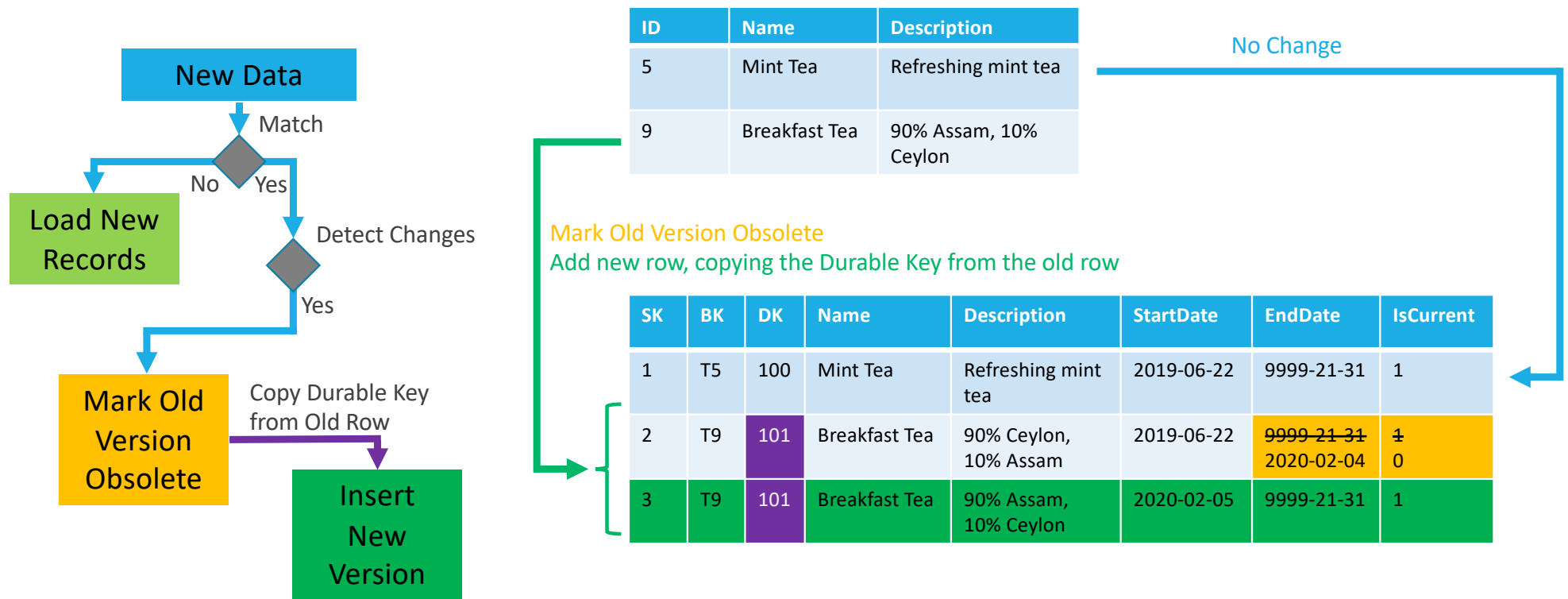 design pattern
- This is the SCD type 7!

SCD Type 2 Example

New Data

Match

No — Yes

Load New Records

Detect Changes

Yes

Mark Old Version Obsolete

Insert New Version

ID	Name	Description
5	Mint Tea	Refreshing mint tea
9	Breakfast Tea	90% Assam, 10% Ceylon

No Change

Mark Old Version Obsolete
Add new row

SK	BK	Name	Description	StartDate	EndDate	IsCurrent
1	T5	Mint Tea	Refreshing mint tea	2019-06-22	9999-21-31	1
2	T9	Breakfast Tea	90% Ceylon, 10% Assam	2019-06-22	~~9999-21-31~~ 2020-02-04	~~1~~ 0
3	T9	Breakfast Tea	90% Assam, 10% Ceylon	2020-02-05	9999-21-31	1

SCD Type 2 With Durable Key Example

New Data

Match

No Yes

Load New Records

Detect Changes

Yes

Mark Old Version Obsolete → Copy Durable Key from Old Row → **Insert New Version**

ID	Name	Description
5	Mint Tea	Refreshing mint tea
9	Breakfast Tea	90% Assam, 10% Ceylon

No Change

Mark Old Version Obsolete
Add new row, copying the Durable Key from the old row

| SK	BK	DK	Name	Description	StartDate	EndDate	IsCurrent
1	T5	100	Mint Tea	Refreshing mint tea	2019-06-22	9999-21-31	1
2	T9	101	Breakfast Tea	90% Ceylon, 10% Assam	2019-06-22	~~9999-21-31~~ 2020-02-04	~~1~~ 0
3	T9	101	Breakfast Tea	90% Assam, 10% Ceylon	2020-02-05	9999-21-31	1

Implementation Tips & Tricks

- Surrogate keys assigned using IDENTITY as usual
- Durable keys assigned
  - From a SEQUENCE object
  - Using a DEFAULT VALUE constraint
  - The default value can be overridden unlike an IDENTITY column
  - Uniqueness at the entity level is implemented using a UNIQUE INDEX, with a filter on the IsCurrent column to only include current rows
  - A trigger can be used to ensure the durable key is not updated
- A simple view is added with `where IsCurrent = 1` column to provide the equivalent of an SCD type 1
- Fact tables can add the durable key as a physical column to join to the view, or table
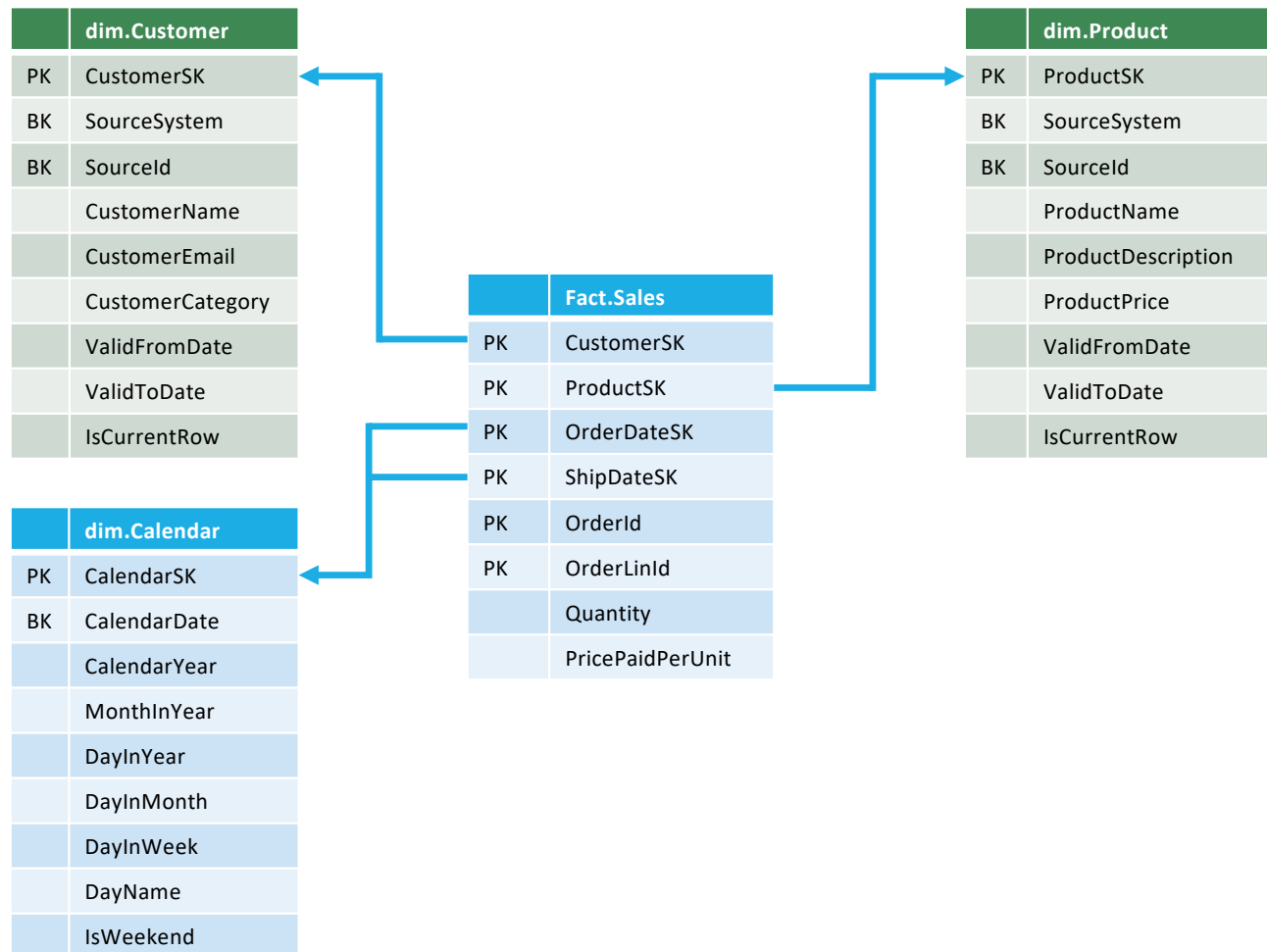
Summary

- Pattern based approach
- Simplify SCD Type 2 queries by changing to SCD Type 7
  - Easy development
  - Better user experience
  - Simpler, more performant queries
- Allows more advanced querying
  - Point in time views of dimensions, like temporal tables
  - Window functions can be used to analyse status changes, eg sales pipelines
- Easy to setup and backport to existing solutions

Resources

- Demos & Slides

  https://github.com/NJLangley

- Other links

  https://www.kimballgroup.com/2012/07/design-tip-147-durable-super-natural-keys/

  https://www.kimballgroup.com/data-warehouse-business-intelligence-resources/kimball-techniques/dimensional-modeling-techniques/type-7/

  http://www.andrewleesmith.co.uk/2017/02/17/updating-type-2-scd-attributes-using-the-t-sql-merge-statement/

  https://rittmananalytics.com/blog/2020/1/2/modelling-slowly-changing-dimensions-type-23-and-6-using-dbt-and-looker